Learning Shape from Images



Olivia Wiles

St Peter's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Supervised by Professor Andrew Zisserman

Trinity Term 2020

This thesis is submitted to the Department of Engineering Science, University of Oxford, in fulfilment of the requirements for the degree of Doctor of Philosophy. This thesis is entirely my own work, and except where otherwise stated, describes my own research.

Olivia Wiles, St Peter's College

Acknowledgements

This thesis would not have been possible without the support and guidance of my supervisor, Professor Andrew Zisserman. He has been a great pleasure to work with over the last four years, and I have benefited greatly from his wide knowledge, research ideas, and presentation advice. I have grown and learned immensely with his help.

I have also had many wonderful collaborators: in particular Sophia Koepke and Sebastien Ehrhardt. I have learned from useful conversations with them which pushed our work forward. I am also grateful to the people at FAIR: Justin Johnson, Georgia Gkioxari, and Richard Szeliski. They made my internship project enjoyable and gave helpful, detailed insights. Finally, I thank my colleagues in VGG, during my Facebook internship and in the wider computer vision community who were not directly related to my work but still provided advice, appreciation, coffee breaks, feedback, and guidance.

In addition to my colleagues, I want to thank my housemates and friends from the past four year who were fun to forget about DPhil worries and frustrations with. I also thank BuzzGym (along with Erika Lu) and BaryUKs Instagram videos which provided necessary exercise diversion.

I thank my parents for their infinite support and patience throughout the DPhil: listening to the frustrations and failures along the road while cheering the successes.

Finally, I would like to thank EPSRC for funding my DPhil.

Olivia Wiles
St Peter's College

Doctor of Philosophy
Trinity Term, 2020

Learning Shape from Images

Abstract

This thesis explores how to harness neural networks to learn 3D structure from visual data. Being able to estimate 3D structure is of vital importance in many applications such as VR/AR, medical imaging, and computational photography. Recently, neural networks have been shown to be effective at learning complex functions from data, including in a 3D setting. The central question this thesis addresses is how to overcome the limitations inherent in a strongly supervised application of neural networks using given 3D information. There are three tasks we explore that require 3D understanding or are a step in a larger 3D system: (i) correspondence estimation, (ii) single image 3D prediction, and (iii) view synthesis.

A key tool used throughout this thesis is *self-supervision*. The core idea of self-supervision for learning about 3D structure is to train models on varied and challenging datasets *without* requiring knowledge of the true, underlying 3D structure.

The first task considered is how to improve correspondence estimation between images. Instead of following standard approaches that condition descriptors on a single image, we use efficient, complex neural networks to condition the descriptors on both images and achieve state of the art or comparable results.

The second task, to which we devote the majority of this thesis, is how to estimate and represent 3D structure from a single image *without* requiring knowledge of the true 3D structure. We explore three methods for predicting 3D information using a self-supervised training approach; these methods vary in terms of the domain, self-supervised setup and 3D representation. We first use silhouettes and depth to train an embedding to represent 3D structure on a dataset of realistic sculptures that can incorporate non photometrically consistent input views. Being able to incorporate images of the same object with different materials is not feasible with traditional approaches. We then use geometric correspondences to train a single image depth predictor on a dataset of real images of sculptures. Training a single image depth predictor is challenging using standard methods, as the images vary dramatically in terms of the context (for example, whether taken at day or night, or in autumn or winter), illumination, and pose. Finally, we use videos of people (or another object class) moving over time to learn class based embeddings that encode the pose and structure of the corresponding instances of the class. This embedding is then used to predict object specific properties (such as landmarks or expression) and achieves high quality results for a variety of object classes, including humans, faces, and animals.

The final task focuses on how to apply self-supervision to a downstream task, view synthesis, which requires implicitly understanding 3D structure. Given an image of a scene or a face, the task is to generate a new view of that scene or face. For a scene, we generate new images according to an input viewpoint change. For faces, we generate new images by manipulating the pose or expression of the face according to another input face or another modality (such as pose or audio). These self-supervised methods achieve high quality results on challenging domains by implicitly learning about the 3D structure of a scene, *without* knowing the true 3D structure.

Contents

1 Introduction **1**

1.1 Correspondence Estimation by Conditioning on Both Images 4

1.2 Learning 3D Without a 3D Dataset 5

1.3 A Downstream Application: View Synthesis 7

1.4 Publications 8

2 Literature Review **11**

2.1 3D structure from a Single Image 11

2.1.1 Shape from X Without Learning 11

2.1.2 Explicit 3D Shape Prediction 12

2.1.3 Self-Supervised 3D Prediction 13

2.2 3D Shape from Multiple Images 16

2.2.1 Structure from Motion (SfM) 16

2.2.2 Representing 3D Structure from Multiple Images 19

2.3 Morphable Models 20

2.4 Applications of Neural 3D 22

2.4.1 View Synthesis from Multiple Images 23

2.4.2 View Synthesis from a Single Image 23

2.4.3 Manipulating a Face in an Image 24

2.4.4 Further Applications 24

3 D2D: Learning to find good correspondences for image matching and manipulation **26**

4 Learning to Predict 3D Surfaces of Sculptures from Single and Multiple views. **51**

5 3D Surface Reconstruction by Pointillism **74**

6 Self-Supervised Learning of a Facial Attribute Embedding from Video 106

7 Self-Supervised Learning of Class Embeddings from Video 128

8 X2Face: A Network for Controlling Face Generation using Images, Audio, and Pose Codes 141

9 SynSin: End-to-end View Synthesis from a Single Image 166

10 Conclusions and Future Work 189

10.1 Conclusions and Achievements 189

10.2 Possible Extensions 192

10.3 Open Questions in Learning Shape 194

Bibliography 196

Abbreviations

AR ..... Augmented Reality

CNN ..... Convolutional Neural Network

DNN ..... Deep Neural Networks

DoG ..... Difference of Gaussians

MLP ..... Multi Layer Perceptron

MM ..... Morphable Model

MVG ..... Multi View Geometry

MVS ..... Multi View Stereo

RANSAC ..... Random Sample Consensus

SfM ..... Structure from Motion

SLAM ..... Simultaneous Localisation and Mapping

VR ..... Virtual Reality

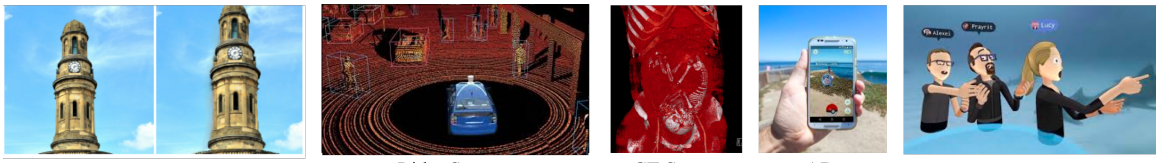
Chapter 1

Introduction

Understanding the underlying 3D structure of visual data is one of the grand challenges in computer vision. Solving this task for arbitrary images and videos would unlock exciting, important applications. For example, representing 3D structure is pivotal for medical imaging, augmented and virtual reality (AR/VR), computational photography, and preserving and virtually disseminating cultural heritage sites, see Fig. 1.1.

While seemingly easy for a human to understand the 3D structure within an image, this is a highly ambiguous and unconstrained problem. Predicting the underlying 3D shape from an arbitrary image is challenging due to the loss of information when projecting from 3D to 2D, previously unseen shapes, and the diversity of scenes to be modelled, as shown in Fig. 1.2. Despite these challenges, the human visual system is able to do this effortlessly by using information implicit in the scene. There are many such information cues including occlusion, perspective, relative size, texture, and shading to name a few. How best to leverage these cues automatically to extract 3D shape from one or more images is an open problem.

There have been a wide range of interesting and successful approaches. These approaches vary from focussing on one cue (for example, only texture), to learning



Computational Photography. 3D Ken Burns. Niklaus et al. SIGGRAPH 2019

Lidar Sensor. Driverless Car. (shorturl.at/szO69)

CT Scan. Wikipedia.

AR. Pokemon Go.

VR. Tech Crunch.

Figure 1.1: **Sample applications using 3D.** 3D is required for autonomous driving, medical imaging, VR/AR, and computational photography.



Figure 1.2: **Images demonstrating the challenges of extracting 3D shape.** Modelling the 3D structure of an image is an inherently challenging and ambiguous problem. Images may vary dramatically in type (for instance the scene may be of a landscape, sculpture, or room), style (such as evening, snowy, or daytime), and the objects may be abstract.

these cues automatically (using neural networks), to using multiple images in order to enforce a stronger set of constraints. The following will discuss traditional approaches to modelling 3D structure in a single or multiple images before discussing how modern methods have transformed the research landscape using neural networks in terms of how to estimate the 3D structure and how to represent 3D structure for downstream applications.

When extracting 3D from a single image, traditionally one had to impose constraints on the image. Methods such as Witkin [1981], Zhang et al. [1999] looked at how to use one cue (such as shading or texture) in order to extract 3D information. However, this is limited to simplistic scenes and does not work for the complex scenes visualised in Fig. 1.2. A notable exception is when the image is from a specific object class, such as faces, for which Blanz and Vetter [1999] build a model to capture the variations of the class. These morphable model (MM) approaches (Fig. 2.3 in Section 2.3) have had great success in modelling and manipulating a single object class (for instance faces or humans). However, they can only be applied to the given object class.

If one can assume multiple images of the underlying 3D scene are available, then stronger constraints can be imposed using multi-view geometry (MVG). A standard structure from motion (SfM) pipeline [Schönberger and Frahm, 2016] operates by first finding correspondences between images and then triangulating matched points in 3D, as discussed in Hartley and Zisserman [2000]. This only generates a sparse reconstruction, which is used to initialise an algorithm [Furukawa and Ponce, 2009, Koch et al., 1998, Schönberger et al., 2016] to extract dense 3D structure. The results

using SfM are impressive, especially for large image collections, allowing for the large scale reconstruction of scenes from thousands of Internet images in Agarwal et al. [2009], as shown in Fig. 2.2 in chapter Section 2.2.1. If real-time performance is required (as is necessary in an AR setting or on device), then more efficient SLAM algorithms can be utilised [Klein and Murray, 2007, 2009]. However, SfM has a number of known limitations and failure cases: for example, SfM requires many images to triangulate points reliably and matches are found primarily in regions of high texture, as they must be reliably detected.

The limitations in traditional approaches led to a surge of interest in how to leverage deep learning to solve these problems. Neural networks have been found to achieve impressive results on a variety of classification and regression tasks [He et al., 2016, Krizhevsky et al., 2012, Simonyan and Zisserman, 2015]. They typically operate as follows: one defines a task to be solved (say image classification), one then collects a large dataset of images and corresponding labels and finally one trains a neural network variant to ingest an image and predict the corresponding known label. A straight-forward application of these approaches into 3D is to obtain a dataset of images and corresponding 3D objects or depth maps and then train a neural network to regress the 3D information. The model should implicitly learn the cues that give clues as to the underlying 3D structure in the image: for example, perspective, occlusion, or others.

While this does indeed work, a standard neural network approach suffers from two challenges: *data collection* and *generalisation*. It is expensive to collect ground truth depth maps for real images, requiring special cameras or equipment, motivating the use of unrealistic synthetic objects and images. Notable exceptions are Li and Snavely [2018], Li et al. [2019], which use SfM to automatically obtain a large scale set of depth maps for a corresponding and diverse set of real images. An additional challenge is that of generalisation: the models only generalise to images from the same domain as that of the dataset the model was trained on. For example, if the model was trained only on road images, the model will only generalise to road images. There are more subtle domains, including the type of camera used, in particular the focal length, as shown by He et al. [2018]. As a result, a difficulty of these approaches is how to obtain a dataset that is as varied and challenging as possible.

These 3D predictions, though, are often not the desired final output of a system. Instead they are a first component in a system that aims to perform some downstream application. These range from generating impressive time lapses [Martin-Brualla et al., 2015a,b, Matzen and Snavely, 2014], meaningfully moving through a set of

images [Snavely et al., 2006], synthesising new views of a scene or image [Agarwal et al., 2009, Blanz and Vetter, 1999], or manipulating the style or rendering of an image [Meshry et al., 2019, Nguyen-Phuoc et al., 2018, Thies et al., 2020]. However, if obtaining 3D structure is an underlying component, then the final result is dependent on the quality of that 3D prediction.

In this work, we consider how to overcome the limitations discussed above. The primary underlying questions we address are to what extent are the underlying assumptions necessary: (i) do we need to consider images independently in a SfM pipeline when obtaining correspondences (Section 1.1)?, (ii) can we leverage more data by *not* requiring a dataset of pairs of images and corresponding 3D shape when training models to predict 3D structure (Section 1.2)?, and (iii) do we need the initial 3D prediction for the downstream application or can we learn a model *directly* for the downstream task (Section 1.3)?

1.1 Correspondence Estimation by Conditioning on Both Images

When constructing the 3D structure of a scene using a SfM pipeline, correspondence estimation is a pivotal component of the pipeline. Correspondence estimation is also central to a variety of other problems including optical flow estimation as in Lucas and Kanade [1981], tracking, and video understanding. There are three main phases of a SfM pipeline: (1) *detect*: reliably detectable regions are extracted, (2) *describe*: descriptors are obtained for these regions, and (3) *correspondence*: the descriptors are matched across images. In particular, the descriptors are conditioned on a single image. This improves efficiency, as the descriptors can be precomputed and saved. However, it means that descriptors must be invariant to changes that may occur in natural images, such as rotation or changes in illumination.

With the advent of neural networks and powerful GPUs, it is possible to run large neural networks efficiently for many images. Due to this efficiency, we are able to explore in chapter 3 whether it is preferable to condition features on *both* images before performing the correspondence operation as opposed to conditioning features on a single image. By conditioning the predicted features on both images, the predicted features (samples visualised in Fig. 1.3) *only* need to be invariant to changes between the images. We explore this idea using a simple model to understand how powerful such an approach can be.

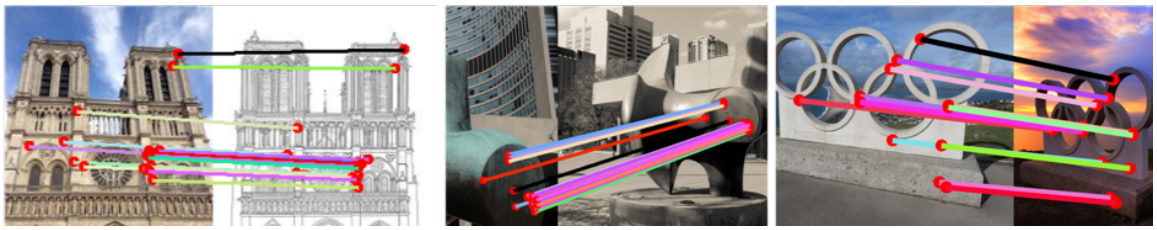


Figure 1.3: **Predicted correspondences.** Correspondences predicted between images of varying style, viewpoint and illumination using our model discussed in chapter 3.

1.2 Learning 3D Without a 3D Dataset

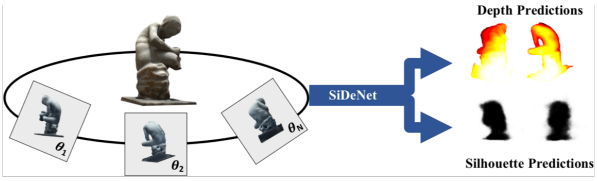
Learning a model to predict 3D structure from an image requires, in a simple approach, a large dataset of image and depth pairs. A straightforward approach of predicting 3D structure from images is to collect a dataset of pairings of 3D objects and 2D images of these objects. There is a large variety of methods in this area, ranging from works predicting depth maps from roads (using a dataset such as KITTI by Geiger et al. [2013]) to works predicting 3D objects using a synthetic dataset of objects (such as ShapeNet by Chang et al. [2015]).

As discussed, requiring a dataset with image and corresponding 3D shape, has two main limitations. First, the data collection step to obtain the initial dataset is expensive; it requires either the use of unrealistic, synthetic data or is limited to a specific domain (such as road scenes). Second, the model will only generalise to new images from the same domain as the training dataset, limiting the utility of models trained on synthetic data or limited domains.

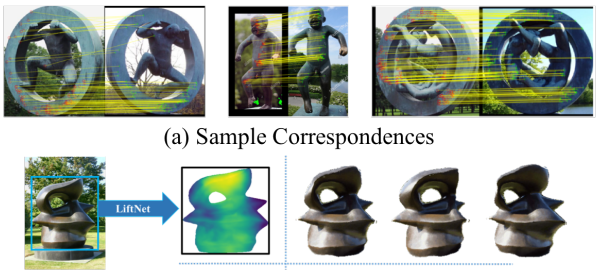
We instead aim to work on realistic objects and 'in-the-wild' real images. As a result, we do not want to require depth for each image; instead we learn the 3D structure by using self-supervision as a tool.

In a self-supervised setting, the aim is to learn useful features directly from data, without ground truth labels. This is done by constructing a proxy task (for example, colourisation in Zhang et al. [2017] or prediction of the future latent space in Oord et al. [2018]), and then tasking the network to solve this proxy task. By choosing the right proxy task, the network must learn to understand meaningful properties of the data; this requires the model to learn useful features. For example, when colourising images, the model must understand semantics in order to predict valid colours: for example, green or red are valid colours for apples but not blue.

In a 3D setting, a self-supervised approach means that we do not use ground truth depth to train the model. Instead the model predicts, from one or more images, a 3D



Chapter 4: Training a multi-view model to predict full 3D structure using silhouettes and depth maps.



(a) Sample Correspondences

(b) Predicted Depth (c) Rendered Views

Chapter 5: Using correspondences to train a single image depth predictor



(A) Sample Videos



(b) Learned Embeddings

Chapter 6,7: Training a class based embedding in a self-supervised manner using video data.

Figure 1.4: Our models that use self-supervision to learn 3D structure.

representation, which is projected into a new view. The proxy task is used to enforce that this projected shape match some known constraint (e.g. that the projected shape must encompass the entire silhouette in the new view). In order to solve this proxy task, the model must learn an implicit representation of 3D shape. Using a self-supervised approach should allow for the use of more data and more varied data, as ground truth 3D structure for each image is not needed.

In chapters 4-7, we use different proxy tasks in order to learn 3D representations using challenging, realistic datasets (visualised in Fig. 1.4). In chapter 4, we use a proxy task based on predicting silhouettes and depth maps in order to learn an embedding that represents the full 3D structure of an object. This embedding can be conditioned on one or more images. A benefit of this approach, as opposed to a traditional method, is that there is no need for the input images to be photometrically consistent: they can be of dramatically different materials with a negligible drop in performance. This presents a significant difference from traditional approaches, in which a large change in material would negatively impact the 3D reconstruction.

In chapter 5, we use a proxy task based on correspondences obtained using MVG. Our insights are that we can (i) obtain a large number of accurate correspondences for pairs of images from data by making use of techniques from MVG and (ii) that these sparse correspondences enforce a strong constraint on the predicted depth in an

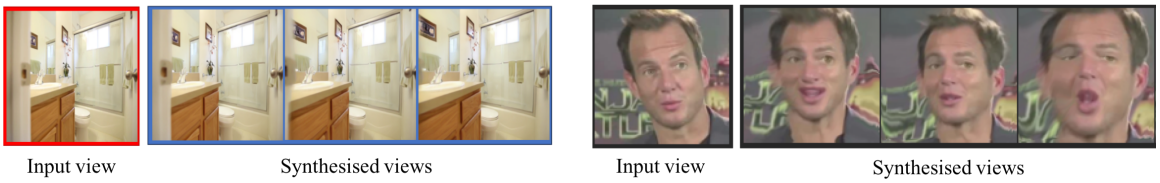
image. Using these correspondences, we train a dense depth predictor by enforcing that a 3D point, projected from the predicted depth map into the new image, matches the true correspondence in the other image. Using a point to point correspondence is a stronger constraint than the silhouette or the standard photometric loss. It can also be used in more general cases: when the object cannot be segmented or the scenes vary too much in illumination, context, or viewpoint, thereby making a photometric loss unsuitable.

In chapters 6-7, we investigate how to use a self-supervised learning task in order to learn a model that represents the variation within an object class. For example, we use our model to learn a representation for faces from a large collection of videos of people giving interviews [Chung et al., 2018]. Using a photometric loss as a supervisory signal, we learn an embedding that encodes class specific properties of a given example (such as the pose and expression of a face).

1.3 A Downstream Application: View Synthesis

While in the previous sections we have considered how to predict 3D structure from a single image, in many scenarios the 3D representation is merely a component in a larger framework. For example, we may want to synthesise a new view from one or more images. We could do this by first predicting the 3D structure and then rendering this 3D information at new, specified viewpoints (the downstream application). For example, other methods have had success at applying a pretrained depth predictor [Niklaus et al., 2019, Shih et al., 2020] or applying a MM face estimator directly to the image [Thies et al., 2016] and then manipulating the extracted 3D structure before refining the prediction.

However, these methods are limited by the quality and robustness of the underlying depth predictor or MM estimator. The depth predictor is trained to predict depth on a given domain (such as road scenes) with image and depth pairs. The depth predictor will fail on new domains (such as indoor environments), and it is trained using a depth prediction loss, not the desired photometric loss for the downstream task. This is not ideal, as the errors arising from predicting incorrect depth may not correlate with the photometric errors arising from projecting the scene using that incorrect depth into a new view (for example, a rough surface predicted as smooth may have minimal depth errors but when reprojected into a new view, the smooth estimate would lead to large photometric errors). The MM estimators traditionally used keypoints in order to perform fitting [Blanz and Vetter, 2003], and so



Chapter 9: Learning a model to synthesise new views of a scene.

Chapter 8: Learning a model to synthesise new views of a face.

Figure 1.5: Samples results from our view synthesis models on realistic scenes or faces.

they fail if the keypoint detection step fails (which is often true for blurry images or extreme viewpoints). While this fitting step has now been updated to use a neural network in Tewari et al. [2017], Tuan Tran et al. [2017], again the generated results can only be as good as the fit of the morphable model.

To address this limitation, we use self-supervision to learn a model to directly solve the downstream application, *without* knowledge of the ground truth 3D structure of the underlying scene or object. Using realistic data, we learn models that ingest a single image and can generate new poses of the given scene or face (see Fig. 1.5). The benefits of learning directly for the downstream task are that we can train directly on realistic data and we are not limited by the quality or robustness of the 3D prediction step. Instead, the implicitly learned 3D representation needs to only be as good as required for the downstream task.

In chapter 8, we consider how to synthesise new images from a single image of a face. Given a source face and a driving face, the model manipulates the source face to have the pose and expression of the driving face. The trained model can then be used to manipulate the face with other modalities, such as audio or pose information.

In chapter 9, we consider how to learn a model for performing view synthesis from a single image. We do this by using a latent point cloud of features, which is trained on a large set of realistic images without knowledge of the true 3D structure of the scene. The trained model ingests an image of a scene and a desired new viewpoint and generates a new image of the scene at that new viewpoint.

1.4 Publications

Refereed publications used in the following chapters include the following.

Chapter 3:

D2D: Learning to find good correspondences for image matching and manipulation. Olivia Wiles, Sebastien Erhardt, and Andrew Zisserman. *In submission.* 2020.

Chapter 4:
SilNet: Single and Multi-view Reconstruction by Learning from Silhouettes. Olivia Wiles and Andrew Zisserman. *In British Machine Vision Conference (BMVC).* 2017.

An extended version was published as a journal article.

Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views. Olivia Wiles and Andrew Zisserman. *In International Journal of Computer Vision (IJCV).* 2019.

Chapter 5:
3D Surface Reconstruction by Pointillism. Olivia Wiles and Andrew Zisserman. *In European Conference on Computer Vision Workshops (ECCVW).* 2018.

Chapter 6:
Self-supervised Learning of a Facial Attribute Embedding from Video. Olivia Wiles*, A Sophia Koepke*, and Andrew Zisserman. *In British Machine Vision Conference (BMVC).* 2018.

Chapter 7:
Self-supervised Learning of Class Embeddings from Video. Olivia Wiles, A. Sophia Koepke, and Andrew Zisserman *In International Conference on Computer Vision Workshops (ICCVW).* 2019.

Chapter 8:
X2Face: A Network for Controlling Face Generation using Images, Audio, and Pose Codes. Olivia Wiles*, A. Sophia Koepke*, and Andrew Zisserman. *In European Conference on Computer Vision (ECCV).* 2018.

Chapter 9:
SynSin: End-to-end View Synthesis from a Single Image. Olivia Wiles, Georgia Gkioxari, Richard Szeliski, and Justin Johnson. *In Computer Vision and Pattern Recognition (CVPR).* 2020.

*Denotes equal contribution

Integrated thesis. This is as an integrated thesis[†]: the chapters are given in the format in which the corresponding papers were published or submitted for publication.

[†] https://www.mpls.ox.ac.uk/graduate-school/information-for-postgraduate-research-students/submitting-your-thesis

Chapter 2

Literature Review

This thesis is primarily focused on how to represent and model 3D structure in a deep learning setting. 3D estimation has been a fundamental component of computer vision since its inception. How to model 3D has been heavily researched, with many successful methods and applications. Here we review those methods most pertinent to our work. These include 3D from a single or multiple images and applications of 3D in a modern setting.

In Section 2.1, we discuss approaches, both modern and traditional, for obtaining 3D information from a single image. In Section 2.2, we review traditional approaches for obtaining 3D structure from multiple images, for example by using SfM, and how this has been updated in a modern framework. In Section 2.3, we explore methods that obtain 3D structure from a single image by using a prior on the object categories to be modelled. Finally, in Section 2.4, we consider some interesting applications using 3D information that have become possible using modern learning frameworks.

2.1 3D structure from a Single Image

Estimating 3D structure from a single image is highly ambiguous, so methods must make use of information cues within an image in order to obtain 3D structure. Traditionally, the community used hand crafted constraints based on known cues. However, with the ability of neural networks to learn complex functions, modern methods typically use a neural network to learn these information cues.

2.1.1 Shape from X Without Learning

There has been a wide range of approaches to extract 3D shape from a single image using information cues. These cues include shading, texture, the object class (please

refer to Section 2.3), human interaction, or knowledge of the illumination. Shape from shading, reviewed by Zhang et al. [1999], has had limited success at reconstructing scenes beyond refining a scene obtained via MVS, as in Samaras et al. [2000]. Shape from texture [Blake and Marinos, 1990, Gårding, 1992, Malik and Rosenholtz, 1997, Witkin, 1981] uses the variation of a pattern over the surface to reconstruct the surface. While useful in a laboratory environment, it has limited application to 'in-the-wild' images. Shape from human interaction uses a human to align a 3D model to an image [Kholgade et al., 2014] or to sketch an object in 3D [Chen et al., 2013]. Finally, shape from illumination models the components of image formation (shape, albedo, and reflectance) in order to extract 3D shape [Barron and Malik, 2011, 2012].

2.1.2 Explicit 3D Shape Prediction

The traditional shape from X approaches work primarily on constrained domains. The hope with neural networks is to be able to leverage the complex information cues that exist in the real world (such as occlusion, perspective, and others) automatically with a neural network by learning on large amounts of data. There are two main general approaches: to predict depth for 'in-the-wild' images or to predict the full 3D shape of an object, which requires hallucinating parts of the object unseen in the image. There are two main challenges: how to represent the 3D structure and how to obtain the ground truth pairings between images and corresponding 3D data.

Learning to predict the 3D shape of objects. Given an image of an object, the aim is to predict the full 3D object. The main dataset used for this task is ShapeNet by Chang et al. [2015], which consists of synthetic objects from 20 categories. These are then rendered to create image and corresponding 3D object pairs. The main research question has been how to represent 3D objects within a neural network such that high fidelity details can be captured. Initially, voxels [Choy et al., 2016, Girdhar et al., 2016, Häne et al., 2017, Riegler et al., 2017] were used because 2D convolutions can be easily extended into 3D convolutions. However, voxels are computationally expensive, limiting the level of detail that can be expressed. As a result, subsequent methods used templates [Tulsiani et al., 2017a], 2.5D representations [Wu et al., 2017], point clouds [Fan et al., 2017], atlas maps [Groueix et al., 2018], meshes [Gkioxari et al., 2019, Wang et al., 2018], and most recently implicit surfaces [Mescheder et al., 2019, Michalkiewicz et al., 2019]. The benefit of using an implicit surface as opposed to the other methods is that it can cover the full surface (unlike a point cloud), it

encodes the surface manifold (unlike point clouds) and produces high quality, smooth results without requiring regularisation terms (unlike meshes).

Learning to predict 'in-the-wild' depth maps. While impressive 3D reconstructions can be achieved using the aforementioned approaches, there is a large domain gap between the synthetic images on which these systems operate and the wider domain of generic images.

In order to bridge this gap, methods have considered how to predict depth from a single image. This was originally done using a CNN and removing scale ambiguity in the depth prediction by Eigen et al. [2014], but later methods such as Laina et al. [2016] demonstrated that using more complex architectures can further improve performance. However, these original approaches primarily focused on two specific domains: indoor scenes [Silberman et al., 2012] or road scenes [Geiger et al., 2013].

Later work focused on collecting datasets of 'in-the-wild' images that were more varied and diverse. These datasets include images annotated with ordinal information [Chen et al., 2016a], datasets obtained from SfM [Li and Snavely, 2018, Li et al., 2019] and datasets obtained using 3D videos [Ranftl et al., 2019]. These datasets (see Fig. 2.1) have been a driving force in improving 3D prediction for generic images, allowing a wide range of applications.

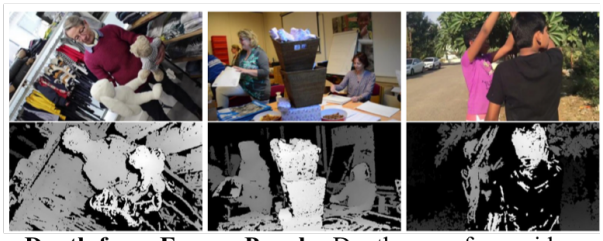
2.1.3 Self-Supervised 3D Prediction

Another approach to dealing with 'in-the-wild' images is to forego the explicit supervision on the 3D representation and instead use self-supervision or indirect supervision in the form of a photometric, keypoint, or silhouette loss.

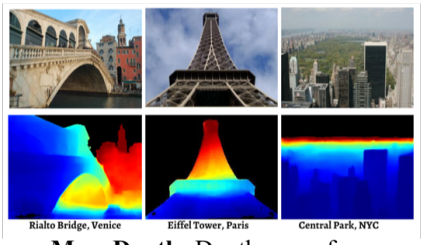
Self-supervised learning seeks to learn descriptors from large amounts of data. They operate by constructing a proxy task (such as colourising an image) that is simple to define on image datasets such that in order to solve the proxy task, the model must understand the objects and properties of the image (for example, apples are green or red). The hope is that by leveraging vast amounts of data, better features will be learned, as they are not limited by the data collection step. Recent methods have demonstrated that it is possible to apply self-supervised learning with current large datasets and networks [Doersch et al., 2015, Hinton and Salakhutdinov, 2006, Noroozi and Favaro, 2016, Oord et al., 2018, Pathak et al., 2016, Vincent et al., 2010, Zhang et al., 2017] and surpass current state-of-the-art supervised models [He et al., 2020, Hénaff et al., 2019].



ShapeNet: Synthetic objects



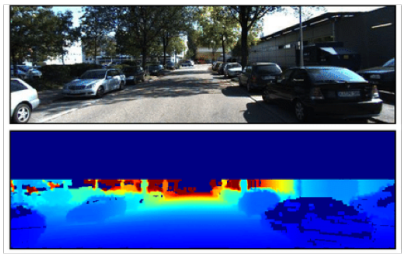
Depth from Frozen People: Depth maps from video data of people doing the mannequin challenge



MegaDepth: Depth maps for landmarks obtained using SfM



Single Image Depth in the Wild: Ordinal labelling of pairs of points as nearer or further away



KITTI: Road data obtained via LIDAR



3DVideos2Stereo: 3D obtained from 3D videos

Figure 2.1: **Sample training information for 3D datasets.** This figure visualises a typical example from each dataset, demonstrating their corresponding limitations (for example, ShapeNet by Chang et al. [2015] is synthetic and KITTI by Geiger et al. [2013] of only road scenes) or benefits (for instance, MegaDepth by Li and Snavely [2018] and Depth from Frozen People by Li et al. [2019] are of varied, challenging scenes, as is 3DVideos2Stereo by Ranftl et al. [2019]).

When applied in a 3D setting, the term self-supervised refers to the fact that the model does not have knowledge of the image's true 3D structure at training or at testing time. These methods vary in terms of how they represent the underlying 3D object or scene, and what proxy task is used to train the model.

When originally applied to objects, methods such as those by Rezende et al. [2016], Yan et al. [2016] took inspiration from work on Visual Hulls [Laurentini, 1994], which formalises how silhouettes carve out the underlying 3D structure. The modern methods used silhouettes in order to train a neural network to regress a latent 3D voxel representation. However, as only the silhouette is used, these methods cannot model concavities on the object. Moreover, the voxel representation is computationally expensive. As a result, later work such as that of Tulsiani et al. [2017b] used photometric losses to train the implicit 3D representation and used more computationally inexpensive representations such as embeddings [Dosovitskiy et al., 2015], point clouds [Insafutdinov and Dosovitskiy, 2018], meshes [Liu et al., 2019a], or implicit surfaces [Liu et al., 2019b]. However, these methods have been limited to simple, synthetic object categories.

Another vein of work has investigated how to automatically extract depth from a pair of images on real scenes. This was first done on stereo pairs by Garg et al. [2016], Godard et al. [2017]. The basic setup was to predict depth in one view and then use the known camera transformation and predicted depth to project the image into the new view and enforce that the generated image should match the true one at the new view. However, assuming stereo pairs limits the applicability of such models. This has led to further work that generalises the approach to video data by predicting both the pose and the depth together [Gordon et al., 2019, Zhou et al., 2017]. However, these works still require photometric consistency between views, limited viewpoint change, and minimal non-static motion. Additionally, the photometric loss is a weak loss; it does not impose the harder, point to point correspondence of optical flow.

In our work in chapters 4 and 5, we also aim to work on real scenes and challenging, varied domains. However, unlike the works of Garg et al. [2016], Godard et al. [2017], Gordon et al. [2019], Zhou et al. [2017], we do *not* use a photometric loss. The reason for this is two fold. First, one may have multiple views of an object that differ in material but are of the same 3D shape. We want to learn a 3D representation using this data but this is not possible with a photometric loss; hence we use a silhouette and (optionally) a depth loss in chapter 4 to learn the full 3D structure. Second, images of the same object may be taken under dramatically different conditions (for example, on a snowy day versus on a summer's day or at night versus during the day),

illumination, and viewpoint meaning that a photometric loss would give erroneous gradients. As a result, in chapter 5 we use correspondences robustly obtained using MVG in order to provide a more robust point to point loss.

2.2 3D Shape from Multiple Images

Traditionally, the most successful approach for obtaining 3D structure from a large set of unregistered images was to use SfM. Once the images have been registered, MVS can be used to obtain a dense reconstruction. In a modern framework, the pipelines used in SfM and MVS have been updated to use learned components.

2.2.1 Structure from Motion (SfM)

Given many images of a rigid scene, SfM aims to obtain a sparse reconstruction of the 3D scene by simultaneously obtaining the 3D structure and camera poses. This area has a long history in computer vision with many works improving various aspects of the pipeline, as explained or surveyed by Faugeras and Luong [2001], Hartley and Zisserman [2000], Szeliski [2010]. We first describe the typical components of a SfM before describing further steps that refine the prediction to obtain a dense reconstruction. The components and how they work together are illustrated in Fig. 2.2*.

Feature extraction. The first component of a SfM system is to find features for each image, which will be matched to obtain correspondences in the following stage. This was traditionally done in two stages. The first *feature detection stage* finds regions that are likely to be consistently found across different images. Intuitively, we wish to find textured regions that can be found in other images; textureless regions (such as the sky) will have many ambiguous correspondences. To achieve this, many detectors have been suggested, ranging from simple corner detectors such as in Harris and Stephens [1988] to those that are invariant to in-plane rotations using a Gaussian kernel [Förstner and Gülch, 1987, Harris and Stephens, 1988, Schmid et al., 2000], are invariant to affine changes [Mikolajczyk and Schmid, 2004, Mikolajczyk et al., 2005], or use non maximal suppression to find the best detections within a window as in Brown et al. [2005].

*Images are taken from http://cs.brown.edu/courses/cs143/2013/results/proj2/ren/, Snavely et al. [2006], and https://www.csie.ntu.edu.tw/~cyy/courses/vfx/16spring/lectures/handouts/SFMedu_brief.pdf.

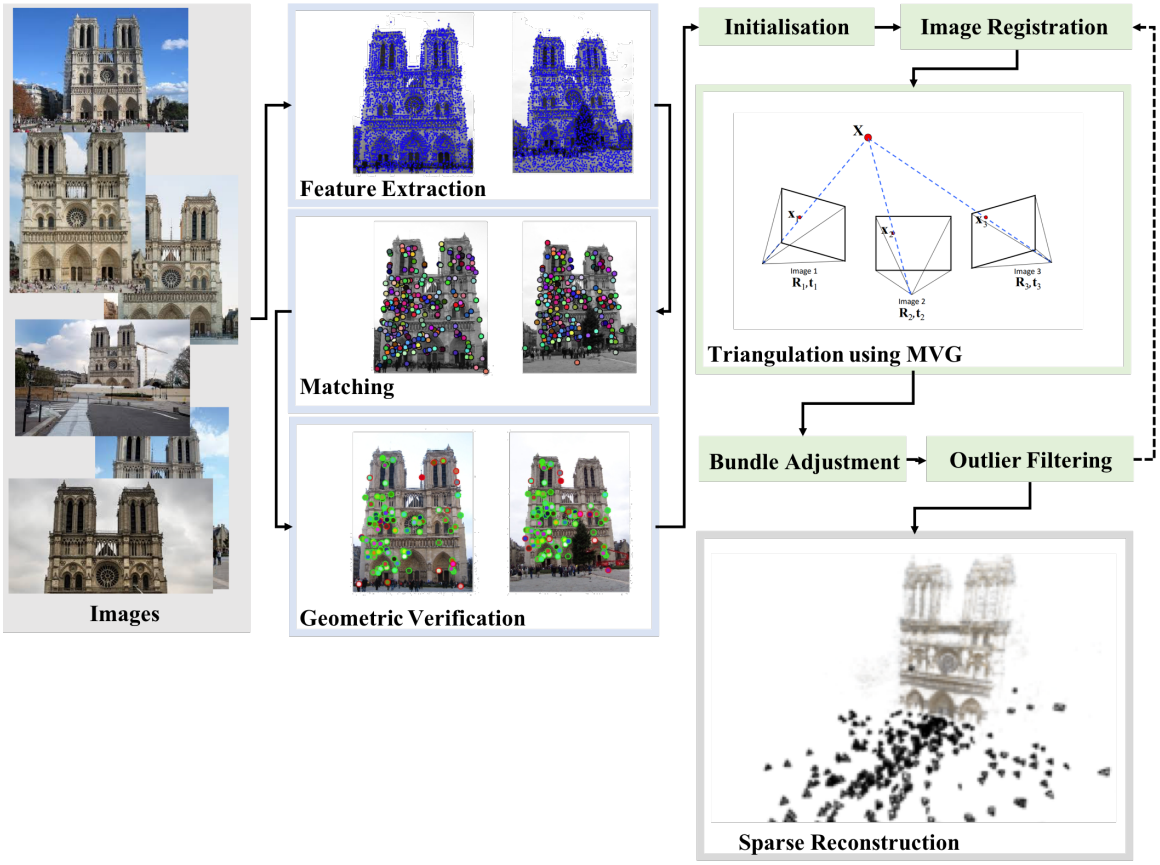


Figure 2.2: **A typical SfM pipeline based on that of Schönberger and Frahm [2016].** Given a set of images, distinctive features are extracted, matched, and verified. They are then used to initialise the 3D reconstruction which uses triangulation to decompose the correspondences into 3D points and cameras. This is used to obtain a sparse reconstruction.

The second *feature descriptor stage* creates a descriptor for the region that is invariant to certain changes. The most widely used descriptor is the SIFT descriptor [Lowe, 2004], which is invariant to rotation, translation, scale, and to some extent illumination. Other descriptors include those of Bay et al. [2006], Calonder et al. [2010], Ke and Sukthankar [2004], Leutenegger et al. [2011], Rublee et al. [2011].

In a modern setting, these stages have been replaced by models trained on data. The hypothesis of these modern approaches is that, due to the traditional methods being hand-engineered, automatically learning reliably detectable regions and invariant descriptors should be better able to automatically extract the required invariance for a domain. These methods focus on automating the detector stage, descriptor stage, or both [Balntas et al., 2016, DeTone et al., 2018, Dusmanu et al., 2019, Ono et al., 2018, Savinov et al., 2017, Simo-Serra et al., 2015, Simonyan et al., 2014, Yi et al., 2016, Zhang and Rusinkiewicz, 2018]. In our work in chapter 4, we forego the detector stage and instead learn descriptors conditioned on *both* images.

Matching. Given a set of features obtained in the previous stage, this stage aims to find matches between pairs of images. This stage operates in two steps. First the features are compared to give a set of likely correspondences. These are then verified using geometric verification as discussed in Hartley and Zisserman [2000]. Traditionally this was done by using explicit geometric verification and a robust algorithm such as RANSAC [Fischler and Bolles, 1981]. However, modern approaches such as those of Brachmann et al. [2017], Moo Yi et al. [2018], Ranftl and Koltun [2018] have demonstrated that a differentiable formulation of geometric verification can be used within a neural network to train the entire pipeline, including the pose estimation, in an end-to-end manner.

Obtaining a sparse reconstruction. Once a number of high quality matches have been obtained, the next step is to use triangulation to obtain the 3D reconstruction [Hartley and Zisserman, 2000]. This typically proceeds by finding a good initial pair of images which are used to triangulate an initial set of 3D points. Further images are incorporated sequentially into this initial model. Finally, bundle adjustment [Hartley and Zisserman, 2000, Triggs et al., 2000] is used to minimise a least squares error and refine the estimated camera parameters and 3D positions. This approach was later scaled up to large scale image collections [Agarwal et al., 2009, Crandall et al., 2011].

Obtaining a dense reconstruction. Given this initial sparse reconstruction, a dense reconstruction can be obtained by by using a MVS algorithm such as that of Schönberger et al. [2016]. While we do not review in depth the large set of approaches used to obtain a dense reconstruction from a set of registered images, we refer the reader to Szeliski [2010] for a review on the different approaches, including local algorithms (based on correlating local regions) and global algorithms (based on optimising a global set of energy terms).

Limitations. While the traditional SfM pipeline is a powerful tool and has been used for a wide range of impressive applications, as reviewed in later sections, it has a set of limitations that the computer vision community has sought to overcome with modern frameworks based on neural networks. We review these modern systems later in this chapter, but here we mention some of the limitations of SfM pipelines that these newer setups seek to overcome.

The most obvious limitation is that a SfM pipeline requires many images of a scene in order to perform the sparse reconstruction. It cannot be run on a single image at test time. The second limitation is that, in order to perform the correspondence estimation, a SfM pipeline requires that the images be taken of the same object with a similar texture. A final limitation of a SfM pipeline is the inability to handle moving objects within the scene; a static scene is assumed.

2.2.2 Representing 3D Structure from Multiple Images

Given multiple images of a scene, these modern methods take inspiration from traditional methods in order to construct a learning pipeline to predict 3D structure. These methods differ in whether they aim to (i) model a full object or (ii) model an arbitrary scene. They also differ in whether they are explicitly supervised with known 3D information or are self-supervised.

One line of work focuses on reconstructing a full object. For example Ji et al. [2017], Kar et al. [2017], Sitzmann et al. [2019a] assume known cameras for the images which are used to combine estimated features from the image along epipolar lines within a voxel grid. The 3D representation is trained using a photometric loss: when the learned 3D voxel representation is projected into a new view, the generated image should match the known image at that view. However, voxels become prohibitively expensive when modelling large scenes. As a result further methods have considered other representations including depth maps [Khot et al., 2019, Yao et al.,

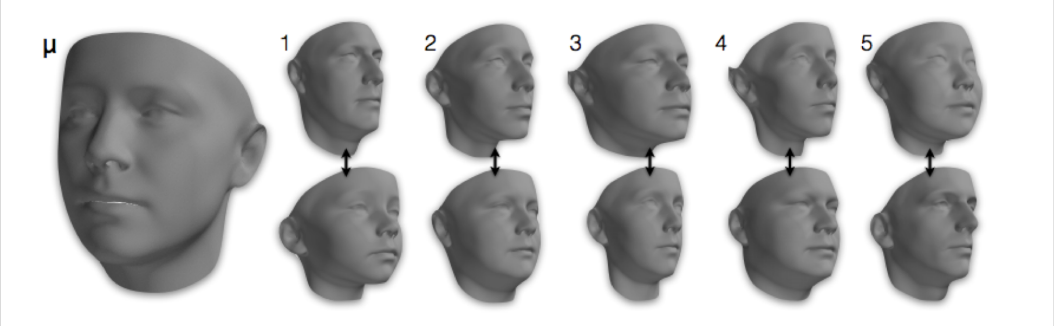
2019] and implicit surfaces [Sitzmann et al., 2019b] in order to model the learned 3D representation.

However, for more general scenes, reconstructing a full object is impractical or ill posed (for example, there is no distinct object in a landscape image). As a result, this line of work takes inspiration from stereo imagery and typically assumes two or more images of a scene in order to obtain the 3D representation [Huang et al., 2018, Kendall et al., 2017, Mayer et al., 2016]. Obtaining such a dataset of depth with real images is challenging, limiting the number and domain of the training images and thereby generalisability of these methods. Consequently, later methods focus on using more general imagery and learning the 3D structure with a self-supervised approach based on a photometric loss. For example, Tulsiani et al. [2018b], Zhou et al. [2018] use pairs of images registered using SfM from videos on YouTube to learn a layered depth representation while Flynn et al. [2016] use a plane sweep volume to represent 3D structure for street scenes.

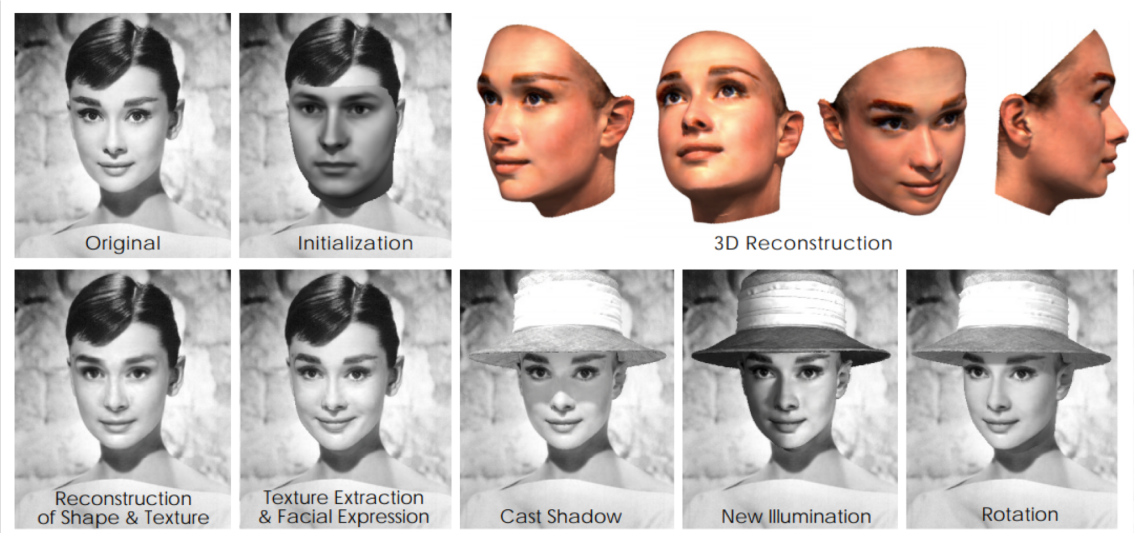
2.3 Morphable Models

If the aim is to model a particular class of objects and reconstruct instances of the class, then one can make use of this information to build more specialised but powerful models. For example, Blanz and Vetter [1999], Booth et al. [2016] learn a 3D statistical model of faces from 3D scans. Related, Debevec et al. [1996] use a basic 3D model of an architecture scene, obtained using user feedback and multiple images of a given architectural example, which is used to render new views and extract additional detail using stereo correspondence.

The statistical models of Blanz and Vetter [1999], Booth et al. [2016] describe the modes of variation and then can be fit to images using analsyis-by-synthesis [Romdhani et al., 2002] or by using auxiliary information, such as keypoints and silhouettes. The fitted model can then be used to manipulate an image (as in Fig. 2.3). Originally, these statistical models were developed for human faces using scanning technology to obtain the 3D reconstructions. Recent work has obtained statistical models for more varied classes such as hands, animals, and the full body [Khamis et al., 2015, Loper et al., 2015, Zuffi et al., 2017] by making use of similar techniques. However, fitting these models still requires optimising over auxiliary information such as keypoints or silhouettes to fit the morphable model to an image (as in Bogo et al. [2016] for human bodies). Obtaining this auxiliary information requires manual annotation of images which is not practical for user focussed applications.



(a) Principal components of the morphable model by Booth et al. [2016]



(b) Fitting a morphable model to an image and manipulating the corresponding image as shown in Blanz and Vetter [1999]

Figure 2.3: **Face Manipulation with a Morphable Model.** (a) To manipulate a face with a morphable model, first a statistical model is created from a large dataset of 3D face scans. This morphable model represents the modes variation present in the underlying class, such as varying face width or varying nose types. (b) As demonstrated by Blanz and Vetter [1999], a morphable model is a powerful tool for manipulating a face in an image. By manipulating the parameters of the morphable model, the illumination, pose and expression of a face can be manipulated, as shown by the Audrey Hepburn example here.

As a result, modern work has considered how to learn to fit these models automatically using a neural network. The aim is to learn a model that, given an image, can automatically determine the parameters of the morphable model. The main challenge is the ability to find matching pairs of fitted morphable models and the corresponding images. A number of successful methods have demonstrated how to use discriminative learning, synthetic data, partial supervision (in the form of keypoints), or photometric supervision in order to learn such a model [Kanazawa et al., 2018a, Pavlakos et al., 2018, Tewari et al., 2017, Tran and Liu, 2018, Zuffi et al., 2019]. However, these approaches still require the underlying 3D MM in order to learn their system. These approaches cannot be applied to domains for which obtaining a 3D MM is impractical. This has led to interesting approaches, such as Kanazawa et al. [2018b], Kar et al. [2015] that learn a deformation model from a mean shape directly on data annotated with silhouettes or keypoints.

While these approaches work well and have clear practical applications, they require auxiliary information in the form of either a MM (which requires scanning a large number of samples of the object class in order to model the modes of variation) or annotated keypoints and silhouettes. We seek instead in chapters 6 and 7 to learn a model that similarly encodes information about the pose and deformation of an object class but *without* requiring a MM or auxiliary information.

A related, new line of work, is to learn a generative model on a class of images. Such generative models can be used to manipulate an object by moving between the latent embedding of the same object in different viewpoints Brock et al. [2019]. By explicitly encoding 3D structure within a model, then the 3D representation can be manipulated to generate new views of a generated sample Nguyen-Phuoc et al. [2019]. These results imply that the model has learned implicitly the 3D properties of an object class. However, this area is nascent and how best to extract 3D structure from a generative model is still an open research question.

2.4 Applications of Neural 3D

While it is important to be able to extract 3D structure, it is also important to note that often the 3D structure is merely an intermediary representation for a downstream task. Using a neural network, the downstream task can be learned automatically and new applications made possible. We include some interesting applications that have become possible or improved using modern neural networks. We focus on those

directly related to our work (view synthesis and face manipulation) but also mention interesting, related applications, some of which are visualised in Fig. 2.4.

2.4.1 View Synthesis from Multiple Images

The ability to synthesise new views of a scene has been discussed already as a form of supervision for learning depth or a 3D representation. However, for a number of works, the view synthesis task is the main focus of the paper, not the quality of the implicit 3D representation. This task has inspired new representations that can encode and hallucinate high levels of detail.

One line of work interpolates new views of an object given a large number of example images of the object by inferring the underlying 3D geometry. The challenge is to be able to preserve and interpolate high quality detail for a large number of views. Traditionally, this was done using depth maps [Chaurasia et al., 2013, Penner and Zhang, 2017] or multi-view geometry [Debevec et al., 1998, 1996, Fitzgibbon et al., 2005, Kopf et al., 2013, Pollefeys et al., 1998, Zitnick et al., 2004].

In the learning era, DNNs can be used in the following ways to improve the synthesis results. They can be used to improve noisy, incomplete depth prediction for view synthesis [Aliev et al., 2019, Choi et al., 2019, Hedman et al., 2018, Martin-Brualla et al., 2018, Meshry et al., 2019, Novotny et al., 2019]. DNNs can learn an implicit voxel representation of an object from many views [Lombardi et al., 2019, Olszewski et al., 2019, Sitzmann et al., 2019a], which is used to synthesise new views. DNNs can synthesise views from two images within a known, narrow baseline [Flynn et al., 2019, Srinivasan et al., 2017, 2019, Tulsiani et al., 2018b, Xu et al., 2019, Zhou et al., 2018]. Finally, DNNs can learn an implicit representation of a scene from many images which encodes high fidelity details [Mildenhall et al., 2020].

2.4.2 View Synthesis from a Single Image

DNN methods for synthesising new views from a *single* image either rely on a pre-trained depth predictor or learn it in an end-to-end fashion using purely image transformations. Niklaus et al. [2019], Shin et al. [2019], Tulsiani et al. [2018a] rely on large scale benchmarks with image and ground-truth 3D data (and their corresponding annotation effort) to obtain their depth predictors. To not require 3D data, Chen et al. [2016b], Kulkarni et al. [2015], Park et al. [2017], Sun et al. [2018], Tatarchenko et al. [2016], Worrall et al. [2017], Zhou et al. [2016] learn a view synthesis network in an end-to-end manner using purely image to image transformations. Later work

explores performing view synthesis by way of implicitly encoding the 3D structure as an implicit surface or a depth map [Chen et al., 2019, Sitzmann et al., 2019b]. However, these works typically consider constrained, synthetic settings or domains and so do not need to model or hallucinate disoccluded regions for general, real world scenes.

2.4.3 Manipulating a Face in an Image

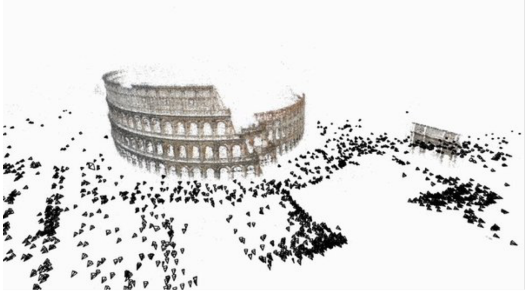
Starting with the seminal work of Blanz and Vetter [1999], there has been research into how best to model and manipulate a face. Traditionally this was done by manipulating the parameters of the morphable model [Dale et al., 2011, Olszewski et al., 2017, Thies et al., 2016, Vlasic et al., 2005] or using a 2D warping method [Averbuch-Elor et al., 2017] based on facial landmarks. It has now become possible to manipulate a face directly by learning in an end-to-end manner. In particular this may be through another face, an emotion encoding, a pose encoding, text, or audio [Chung et al., 2017, Ding et al., 2018, Kulkarni et al., 2015, Suwajanakorn et al., 2017, Tran et al., 2017, Worrall et al., 2017]. These methods can give surprisingly realistic results, leading to ethical concerns but also applications in film and video editing.

2.4.4 Further Applications

Exploring large scale photo collections. The ability to reconstruct a large scale 3D reconstruction, as shown by Agarwal et al. [2009], allows for images to be viewed in a meaningful way, as introduced in Schaffalitzky and Zisserman [2002] and further explored in Snavely et al. [2006].

Time lapses. Another interesting application is the ability to reconstruct cities and automatically extract time lapses, which show how a specific scene or area changes over time [Martin-Brualla et al., 2015a,b, Matzen and Snavely, 2014].

Rendering or stylisation. A final application demonstrated by Meshry et al. [2019], Nguyen-Phuoc et al. [2018], Thies et al. [2020] is to be able to manipulate the style of an image while maintaining the 3D shape of the scene (this is as opposed to stylising using purely image to image transformations) according to a given style: for example, day-time, night-time or other forms of illumination.



(a) Large scale reconstruction by Agarwal et al. [2009] from thousands of photos.



(b) Time lapses of city scenes over time by Matzen and Snavely [2014].



(c) Exploration of photo collections using 3D by Snavely et al. [2006].



(d) Stylisation of a scene using an image by Meshry et al. [2019].

Figure 2.4: **Applications of large scale 3D reconstruction.** Given the ability to reconstruct large scale scenes from thousands of images (a), there are many interesting applications including generating time lapses of city scenes (b), exploring photo collections (c), or stylising scenes using a given image (d).

Chapter 3

D2D: Learning to find good correspondences for image matching and manipulation

This work is currently under submission.

It describes a new approach to extracting correspondences for pairs of images by conditioning the features that are matched on both images, as discussed in Section 1.1. We achieve state of the art or comparable results on a wide range of datasets and tasks, ranging from local matching and camera localisation to 3D reconstruction.

D2D: Learning to find good correspondences for image matching and manipulation

Olivia Wiles
VGG, University of Oxford
ow@robots.ox.ac.uk

Sebastien Ehrhardt
VGG, University of Oxford
hyenal@robots.ox.ac.uk

Andrew Zisserman
VGG, University of Oxford

Abstract

We propose a new approach to determining correspondences between image pairs under large changes in illumination, viewpoint, context, and material. While most approaches seek to extract a set of reliably detectable regions in each image which are then compared (sparse-to-sparse) using increasingly complicated or specialised pipelines, we propose a simple approach for matching all points between the images (dense-to-dense) and subsequently selecting the best matches. The two key parts of our approach are: (i) to condition the learned features on both images, and (ii) to learn a distinctiveness score which is used to choose the best matches at test time. We demonstrate that our model can be used to achieve state of the art or competitive results on a wide range of tasks: local matching, camera localization, 3D reconstruction, and image stylization.

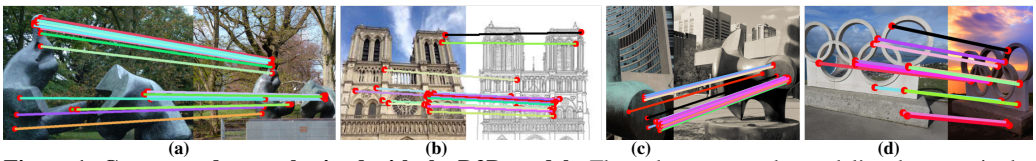


Figure 1: Correspondences obtained with the D2D model. These demonstrate the model's robustness in the face of challenging *scene shift*: changes in illumination (a,d), viewpoint (a-d), context (a,d), or style (b).

1 Introduction

Determining correspondence between two images of the same scene or object is a fundamental challenge in computer vision, important for many applications ranging from optical flow and image manipulation to 3D reconstruction and camera localisation. This task is challenging due to *scene-shift*: a scene dramatically changes over time due to variations in illumination (e.g. day to night), viewpoint, texture, and season (e.g. snow in winter versus flowering trees in spring).

Methods that solve the correspondence task typically follow one of two paradigms: *sparse-to-sparse* or *dense-to-dense*. Under the *sparse-to-sparse* paradigm, most methods follow a *detect-and-describe* approach: first detecting distinctive regions [5, 16, 29, 34, 54] and then extracting descriptors [5, 6, 21, 25, 29, 54] for these regions. These descriptors are then matched between images, often using additional geometric constraints [17]. Recent approaches have sought to learn either or both of these components [3, 8, 9, 41, 58, 63, 64, 75, 77]. These methods typically only find matches at textured locations, and do not find matches over smooth regions of an object. Additionally, finding these repeatable detections with invariance to scene-shift is challenging [2, 55, 60].

Under the *dense-to-dense* paradigm, methods will typically obtain a dense feature map which is compared between images using, optionally, additional spatial and smoothness constraints [7, 10, 28,

Preprint. Under review.

57, 70, 73]. These approaches mostly focus on obtaining optical flow for limited changes in camera within the same video; the scenes have limited scene shift.

We introduce a new approach for obtaining dense correspondences between a *pair* of images. Previous methods (under both paradigms) learn features for each image *without* knowledge of the other image. Thus, their intermediary features must be invariant to changes – e.g. to scale changes, shift, etc. However, as features become more and more invariant, they become increasingly ambiguous to match (if a feature was invariant to everything, it would be confused for everything). We forsake this invariance and instead *condition* the feature maps on *both* images. This allows the features to be modified based on the differences between the images (e.g. a change in global illumination). Traditionally, this was infeasible, but we can learn such a model efficiently using neural networks.

We additionally regress a *distinctiveness score*: the likelihood that a point in one image will uniquely and correctly match to a point in the other image. This score is regressed from the conditioned features, so the likelihoods can model the distinctiveness of a point *given the other image*. At test time the model ingests a pair of images and outputs a set of dense correspondences, together with their distinctiveness scores. The scores can be used to select the best matches.

In summary, we propose a novel approach and network, D2D, a **D**ense-**2**-**D**ense approach for determining correspondences between an image pair. The approach is based on two key insights: (i) **conditioning** learned features on both images, and (ii) learning a **distinctiveness score** for each feature. As will be seen, the model is simple and scalable and eschews a number of techniques used by other methods to improve performance: high dimensional descriptors (we use a 64D descriptor, half the size of the current smallest descriptor), and multiple scales (we only operate at a single scale, whereas other methods use multiple scales).

We apply D2D to a variety of tasks: local matching, camera localisation, 3D reconstruction, and style transfer. We achieve state-of-the-art (sota) or comparable on all tasks despite our simple approach.

2 Related Work

In this section, we review related work on finding correspondences, broken down into three main paradigms: *dense-to-dense*, *sparse-to-sparse*, *sparse-and-then-dense*.

Sparse-to-sparse. Traditional 3D reconstruction and camera localisation pipelines require accurate correspondences across multiple images. Traditional methods solved this by using a detect-and-describe approach: *first* a distinctive region is *detected* in textured regions [5, 16, 29, 34, 54] and *second* a hand crafted descriptor is used to *describe* the region with varying degrees of invariance to scale, illumination, rotation, and general affine transformations [5, 6, 21, 25, 29, 54]. In modern approaches, the descriptor step [3, 13, 63, 64], detector step [58, 77], or both [8, 9, 41, 75] are replaced with a neural network. We forego the detection step and condition features on both images.

Sparse-and-then-dense. Given sparse-to-sparse correspondences, the camera between images can be estimated and a second algorithm, e.g. PatchMatch [4, 61], used to find dense correspondences between the images. If the images have been rectified using multiple-view geometry [17] and have limited *scene shift*, stereo algorithms such as [15, 44, 66, 74] (and reviewed by [67]) can be used to obtain a dense reconstruction.

Dense-to-dense. Other methods consider how to find dense correspondences between two images. How to define correspondence differs based on the method.

Initially, finding dense correspondence meant finding the precise pixel correspondence between two images of the same scene. Traditional approaches [28, 70] generalised sparse descriptors. Modern methods learn to match all pixels between frames of a video [7, 10, 19, 47, 57, 62, 65, 73]. However, these methods assume photometric consistency between frames: their quality degrades the more the frames differ in time, as the pose and viewpoint progressively change. [62] uses additional semantic information and [51] uses pretrained features to find dense matches between images of the same scene to mitigate this problem.

Later, region flow defined correspondence as a match between a pixel and the corresponding region in the other image (e.g. a point on a dog to the entire dog). This is useful if there is large viewpoint change or non rigid motion [24, 71]. Also related are approaches that seek to learn correspondence between similar scenes [28] or instances of the same semantic class [22, 26, 40, 49, 50, 52].

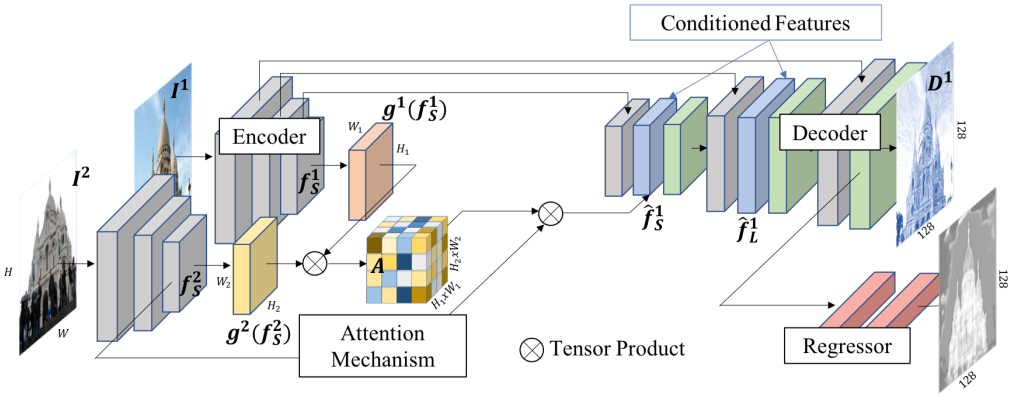


Figure 2: D2D overview for *one* input image. To obtain descriptor vectors D^1 for one input image I^1 conditioned on another I^2, both images (I^1, I^2) are *encoded* via an *attention mechanism*. The conditioned features are then *decoded* to obtain D^1. We also *regress* a distinctiveness mask which is used at test time to ignore unmatchable regions (e.g. the sky or regions visible in only one image). The descriptor vectors D^2 for I^2 are obtained by swapping the input images.

3 Method

Our task is to find dense correspondences between a pair of images of the same scene. The pipeline has two stages. The first stage obtains dense descriptor vectors for each image and a distinctiveness score. The descriptors are conditioned on *both* images so they only have to be invariant to the changes particular to that pair of images. The second stage compares these descriptor vectors to obtain a set of high quality matches. We first introduce our pipeline before describing how it is trained.

3.1 D2D Architecture

Our pipeline for obtaining descriptor vectors and a distinctiveness score for one image I^1 (Fig. 2), is composed of four components. The first component, the **encoder**, projects both images I^1, I^2 to obtain feature maps at two resolutions: f^i_L, f^i_S. The second component, the **attention mechanism**, is used to determine spatial correspondence between the feature maps of the different images. The third component, the **decoder**, uses the attention mechanisms to concatenate the feature maps of the two images; these concatenated features are used to obtain a grid of spatial descriptor vectors D^1 that are conditioned on both images. The final component, the **regressor**, learns a distinctiveness score for each grid position which encodes how likely the match is to be accurate. To obtain descriptor vectors D^2 for the other image, we operate precisely as described above, except that the order of the input images is flipped. This gives a grid of descriptor vectors D^1, D^2 for images I^1, I^2 respectively.

Encoder. Given two images of the same scene, $I^1 \in \mathbb{R}^{H \times W \times 3}$ and $I^2 \in \mathbb{R}^{H \times W \times 3}$, we obtain spatial feature maps: f^i_L and f^i_S at a *l*arger and *s*maller resolution. These will be concatenated within a UNet framework [53] to form the decoder. A CNN with shared parameters is used to encode the images and obtain these spatial feature maps. In practice, we use the feature maps after the last two blocks in a ResNet50 [18] architecture.

Attention Mechanism. We wish to concatenate features from both images in order to condition the model on both input images. However, for a given spatial location, the relevant (corresponding) feature in the other image may not be at the same spatial location. As a result, we use an attention mechanism to model long range dependencies when concatenating features.

An attention mechanism is used to determine where a location i in one set of features g from one image should attend to in another set of features h from another image [71]. For each location i in g, it obtains a feature $\hat{g}_i$ that is a weighted sum over all spatial features in h where A is the similarity matrix comparing g and h using the inner product followed by the softmax normalisation step.

$$\hat{g}_i = \sum_j A_{ij} h_j \qquad A_{ij} = \frac{\exp(g_i^T h_j)}{\sum_k \exp(g_i^T h_k)} \tag{1}$$

To apply this attention mechanism, we operate as follows for f_L^1 (and similarly for f_S^1). As standard, to perform dimensionality reduction the features are projected (with two MLPs $g^1(\cdot), g^2(\cdot)$) and then normalised: $g = \frac{g^1(f_L^1)}{||g^1(f_L^1)||_2}$, $h = \frac{g^2(f_L^2)}{||g^2(f_L^2)||_2}$. The attended features $\hat{f}_L^1$ are then computed as in (1).

Decoder: Conditioned Features. The attended features are incorporated into a UNet [53] architecture in order to obtain a grid of spatial descriptors $D^1 \in \mathbb{R}^{H \times W \times C}$ (see Fig. 2). The attended features $\hat{f}_L^1$ and $\hat{f}_S^1$ are concatenated with the original features and passed through the decoder portion of the UNet. The resulting feature map is $L2$ normalised over the channel dimension to obtain the final descriptors. This ensures that the final descriptors are conditioned on both images.

Regressor: Distinctiveness Score. We regress a distinctiveness score $r(\cdot)_{ij} \in [0, 1]$, for each pixel (i, j), which approximates its matchability and is used at test time to select the best matches. $r(\cdot)_{ij}$ approximates how often the descriptor at (i, j) is confused with negatives in the other image. If it is near 1, the descriptor is unique; if it is near 0, the descriptor is often confused. To regress these values, we use an MLP, $r(\cdot)$, on top of the unnormalised descriptor feature maps.

Determining Matches at Test Time. We want matches at locations k and l in images I^1 and I^2 respectively that are accurate and distinctive (e.g. no matches in the sky). We use the scalar product to compare the normalised descriptor vectors to find the best matches and the distinctiveness score to determine the most distinctive matches. The following similarity score c_{kl} combines these properties such that a value near 1 indicates a distinct accurate match:

$$c_{kl} = r(D_k^1) r(D_l^2) \left[\left(D_k^1\right)^T D_l^2 \right]. \tag{2}$$

Finally, we select the best K matches. We consider all matches which are mutual nearest neighbours: e.g. if the best match for location m in one image is location n in another, and the best match for location n is m, then match (n, m). So if the following holds:

$$m = \texttt{argmax}_j c_{nj} \quad \text{and} \quad n = \texttt{argmax}_i c_{im} \tag{3}$$

These matches are ranked according to their similarity score and the top K used. To select candidate matches, we compare all normalised descriptor vectors in one image to all in the other.

3.2 Training and Loss Functions

Selecting Correspondences at Train Time. Given a ground-truth correspondence map, we randomly select L positive correspondences. For each positive correspondence, we randomly select a large number ($K = 512$) of negative correspondences. These randomly chosen positive and negative correspondences are used to compute both the distinctiveness and correspondence losses.

Correspondence Loss. The correspondence loss is used to enforce that the normalised descriptor maps D^1 and D^2 can be compared using the scalar product to obtain the best matches. At a location i in D^1 and j in D^2 then the standard Euclidean distance metric $d(D_i^1, D_j^2)$ should be near 0 if the corresponding normalised descriptor vectors are a match.

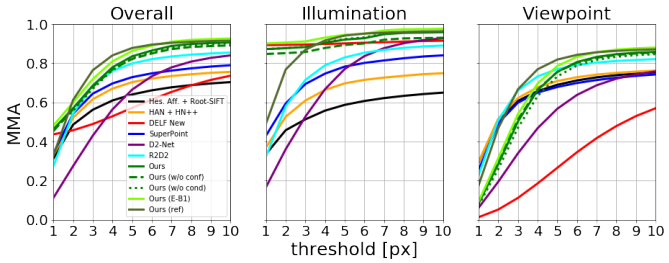
To train these descriptors, we use a standard contrastive hinge loss to separate true and false correspondences (we consider other contrastive losses in the supplementary). For the set $\mathcal{P}$ of L true pairs, the loss $\mathcal{L}_p$ enforces that the distance between descriptors is near 0. For the set $\mathcal{N}$ of LK negative pairs, the loss $\mathcal{L}_n$ enforces that the distance between descriptors should be above a margin M.

$$\mathcal{L}_p = \frac{1}{L} \sum_{(x,y) \in \mathcal{P}} d(D_x^1, D_y^2) \qquad \mathcal{L}_n = \frac{1}{LK} \sum_{(x,\hat{y}) \in \mathcal{N}} \max(0, M + c_x - d(D_x^1, D_{\hat{y}}^2)). \tag{4}$$

$c_x = d(D_x^1, D_y^2), (x, y) \in \mathcal{P}$ re-weights the distance of the false correspondence according to that of the positive one: the less confident the true match, the further the negative one must be from M [9].

Distinctiveness Loss. To learn the $r(\cdot)$ MLP, we need an estimate of how often a descriptor in one image is confused with the wrong descriptors in the other image. Given a set $\mathcal{K}$ of K negative matches in the other image and the margin M, the number of times a descriptor at location x is confused is $m_x = \sum_{\hat{y} \in \mathcal{K}} \mathbb{1}(d(D_x^1, D_{\hat{y}}^2) < M)$. This value is used to regress $r(\cdot)$ which is near 1 if the feature has a unique match (the true match), near 0 otherwise (τ is a hyper-parameter set to $\frac{1}{4}$):

$$\mathcal{L}_r = \frac{1}{L} \sum_{(x,\cdot) \in \mathcal{P}} |r(D_x^1), \frac{1}{(1 + m_x)^\tau}|_1. \tag{5}$$



Method	# Matches
Hes. det. + RootSift [1, 29]	2.8K
HAN + HN++ [35, 36]	2.0K
LF-Net [41]	0.2K
SuperPoint [8]	0.9K
DELF [39]	1.9K
D2 Trained (SS) [9]	2.5K
R2D2 [48]	1.8K
Ours	2.0K

Figure 3: HPatches [2]. Comparison with sota using the mean matching accuracy for different pixel threshold on the HPatches dataset. We also report the mean matches extracted per image pair. For this dataset, one desires more matches with high accuracy. Our method achieves superior performance to all *detect-and-describe* methods when images vary by illumination for all thresholds, and by viewpoint for thresholds $>$ 6px. By a simple refinement strategy (**ours (ref)**), we achieve sota for all thresholds on both viewpoint and illumination.

Training Setup. Our model is trained on MegaDepth [27], which consists of a variety of landmarks, registered using SfM [59]. As each landmark consists of many images taken under differing conditions, we can obtain matches between images that are unmatchable when considered independently.

We train the features end-to-end, but train the distinctiveness score separately by not allowing gradients to flow. In practice we backpropagate on all randomly chosen positive pairs L_p, negative pairs L_n, and additionally the hardest $H = 3$ negative pairs for each positive pair.

The model is trained with a learning rate of 0.0001, the ADAM optimizer [23], a batch size of 16, M=1, and L=512. At train time we use an image size of 256, at test time an image size of 512. We use K=2000 for HPatches and Aachen, and K=8192 when performing SfM. For SfM, we find it is important to use more, rougher correspondences to obtain more coverage in the 3D reconstruction.

4 Experiments

We evaluate the D2D model on three different tasks. The first task directly assesses how well D2D can estimate correspondences between images pairs. The second task uses the D2D correspondences to perform camera localization and obtain high quality 3D reconstructions in challenging situations, with a large amount of *scene shift*. The final task is stylization, and assesses D2D's dense matches.

Ablations. The full model uses the ResNet50 [18] backbone, the attention mechanism and the distinctivness score to reweight matches. We ablate multiple variants. The first (**ours**) is our full model. The second (**ours w/o conf**) is our model without the distinctiveness score. The third (**ours w/o cond**) is our model without conditioning (i.e. the attention mechanism). The final (**ours-E-B1**) is our full model but using an EfficientNet-B1 backbone [68]. This ablation uses a smaller (7M params vs 23M params) and faster (0.7GFlops vs 4.1GFlops) backbone architecture; it is more suitable for practical applications. The supplementary includes further ablations to validate our choices (e.g. the loss function and grid size) and datasets (e.g. YFCC100M [69]) which show our superior results.

4.1 Correspondence Evaluation

We test our model on local matching by evaluating on the HPatches [2] benchmark. We compare to a number of baselines and achieve state-of-the-art results.

HPatches Benchmark. The HPatches benchmark evaluates the ability of a model to find accurate correspondences between pairs of images, related by a homography, that vary in terms of illumination or viewpoint. We follow the standard setup used by D2Net [9] by selecting 108 of the 116 sequences which show 6 images of larger and larger illumination and viewpoint changes. The first image is matched against the other 5, giving 540 pairs.



Figure 4: Aachen Day-Night [55]. Dots represent a point of a correspondence. This visualises the coverage of the correspondences obtained by D2D.

Table 1: Aachen Day-Night [55]. Higher is better. Ours outperforms the other *dense-to-dense (Den)* approach (DenseSFM [55]), and does comparably or better than other sota *detect-and-describe (Spa)* setups. * indicates the method was trained on the Aachen dataset, whereas none of the other methods are.

Method	Type	Threshold Accuracy		
		0.25m (2°)	0.5m (5°)	5m (10°)
Upright RootSIFT [29]	Spa	36.7	54.1	72.5
DenseSFM [55]	Den	39.8	60.2	84.7
Han+, HN++ [35, 36]	Spa	39.8	61.2	77.6
Superpoint [8]	Spa	42.8	57.1	75.5
DELF [39]	Spa	39.8	61.2	85.7
D2-Net [9]	Spa	**44.9**	66.3	**88.8**
R2D2* [48]	Spa	**45.9**	66.3	**88.8**
Ours w/o cond	Den	42.9	62.2	87.8
Ours w/o conf	Den	43.9	64.3	86.7
Ours	Den	**44.9**	**70.4**	**88.8**
Ours (E-B1)	Den	**44.9**	68.4	**88.8**

Evaluation Setup. We follow the evaluation setup of D2Net [9]. For each image pair, we compute the number of correct matches (using the known homography) and report the average number of correct matches as a function of the pixel threshold error in Fig. 3. We then compare to a number of detect-then-describe baselines used in D2Net using their software: RootSIFT [1, 29] with the Affine keypoint detector [33], HesAffNet [36] with HardNet++ [35], LF-Net [41], SuperPoint [8], DELF [39]; as well as to D2Net [9] and R2D2 [48]. These methods vary in terms of whether the detector and descriptors are hand crafted or learned.

Results. As shown in Fig. 3, all variants of our model outperform previous methods for larger pixel thresholds, demonstrating the practicality and robustness of our approach. In comparison to other methods, D2D performs extremely well when the images vary in illumination: it outperforms all other methods. D2D is superior under viewpoint changes for larger pixel thresholds (> 6px). Using the smaller, more efficient (**ours-E-B1**) actually improves performance over the larger ResNet model (**ours**). Using a simple refinement strategy (described in the supplementary) boosts our model's performance under viewpoint changes, giving results superior or comparable to sota methods for all thresholds. Compared to the other datasets we evaluate on, e.g. [55] below, the components of our model have a limited impact at improving performance on the HPatches benchmark, presumably because this dataset has less *scene shift* than the others.

4.2 Using Correspondences for 3D Reconstruction

In these experiments, we evaluate the robustness of our approach on images that vary significantly in terms of illumination and viewpoint, and our model's ability to scale to larger datasets. D2D achieves sota or comparable results on all datasets.

4.2.1 Camera Localization

Aachen Benchmark. In order to evaluate our approach under large illumination changes, we use the Aachen Day-Night dataset [55, 56]. For each of the 98 query night-time images, the goal is to localize the image against a set of day-time images using predicted correspondences.

Evaluation Setup. The evaluation measure is the percentage of night time cameras that are localized within a given error threshold [55]. We use the pipeline and evaluation server of [55] with the matches automatically obtained with our method (Section 3.1). We compare against detect-and-describe methods (RootSIFT descriptors from DoG keypoints [29], HardNet++ with HesAffNet features [35, 36], DELF [39], SuperPoint [8], and D2Net [9]) as well as a dense approach (DenseSFM [55]).

Results. Table 1 shows that our method does better than other *dense-to-dense* approaches and comparably or better than sota detect-and-describe approaches. These results imply that the traditional approach of first finding reliably detectable regions may be unnecessary; using a grid to exhaustively find matches is, perhaps surprisingly, superior in this case. These results also show that our architectural improvements (i.e. using an attention mechanism and distinctiveness score) boost performance and that a more efficient architecture (**Ours-E-B1**) has only a limited impact on performance.

Table 2: SfM. We compare our approach to using SIFT features on 3D reconstruction for two datasets: Large SfM and Sculpture. Both are collections of images from the web containing large variations in illumination and viewpoint. These metrics are a proxy for 3D reconstruction quality, so we encourage the reader to view the reconstructions in the supplementary. X: failure. ↑: higher is better. ↓: lower is better. **blue**: best result.

		Large SfM			Sculpture Dataset								
	Landmark:	MM	Gen	Tow	HM1	HM2	HM3	HM4	HM5	HM6	HM7	HM8	HM9
	# Images:	1344	1463	1576	12	124	250	266	78	31	358	238	74
# Reg. Ims ↑	**SIFT [29]:**	500	1035	804	X	103	**198**	212	61	22	266	**201**	53
	Ours:	**702**	**1072**	**967**	**12**	**108**	194	**215**	**67**	**25**	**284**	**201**	**57**
# Sparse Pts ↑	**SIFT [29]:**	116K	338K	239K	X	48K	70K	102K	28K	9K	128K	**99K**	**23K**
	Ours:	**256K**	**570K**	**452K**	**2.9K**	**63K**	**83K**	**121K**	**40K**	**10K**	**190K**	**99K**	21K
Track Len ↑	**SIFT [29]:**	**6.32**	5.52	**7.76**	X	**5.33**	**5.92**	**5.80**	**4.54**	**4.73**	**4.46**	**5.24**	4.75
	Ours:	6.09	**6.60**	5.82	**3.60**	5.03	5.43	5.61	4.32	4.00	4.23	**5.24**	**4.77**
Reproj Err (px) ↓	**SIFT [29]:**	**0.60**	**0.69**	**0.61**	X	1.31	1.32	**1.28**	1.30	1.33	**1.22**	**1.30**	**1.32**
	Ours:	1.30	1.34	1.32	**1.33**	**1.30**	**1.30**	1.29	**1.29**	**1.26**	1.23	**1.30**	**1.32**
# Dense Pts ↑	**SIFT [29]:**	**1.8M**	**4.2M**	**3.1M**	X	160K	143K	**307K**	73K	**46K**	174K	333K	**54K**
	Ours:	1.1M	2.1M	1.8M	**0.2K**	**188K**	**156K**	296K	**82K**	44K	**187K**	**333K**	53K

4.2.2 Structure from Motion (SfM)

The objective here is to evaluate the correspondences obtained with our model for the task of 3D reconstruction using a smaller or larger number of views. The assessment is on two datasets: first, the standard SfM benchmark that contains many ($\approx$ 1500 images) of the same scene; second, a challenging scenario where the dataset contains fewer ($\approx 10 - 100$ images) images of the same scene and where the object (a sculpture) may differ in material, location, and context.

Baselines. We compare to SIFT [29]. This method first finds repeatably detectable regions for which features are extracted and compared between images. This method works well when there are distinctive textured regions that can be found and matched. Our method, however, aims to find dense matches, so our approach should be more robust when there are fewer textured regions.

SfM Dataset. In this setup, we evaluate on the Local Feature Evaluation Benchmark [60]. This dataset consists of multiple landmarks for which many internet images have been collected. We use Madrid Metropolis (MM), Gendarmenmarkt (Gen), and Tower of London (Tow). Table 2 demonstrates that, using D2D, we consistently register more images and obtain significantly more sparsely reconstructed 3D points (visualisations are given in the supplementary). However, the pixel error is higher and there are fewer dense points. These results imply that our model is more robust for matching and registering challenging images but, presumably due to the grid discretization when creating candidate correspondences, the correspondences are not as pixel accurate, affecting the reprojection error.

Sculpture Dataset. We use images from the Sculpture dataset [12], which consists of images of the same sculpture downloaded from the web. As an artist may create the same sculpture multiple times, a sculpture's material (e.g. bronze or marble), location, or context (e.g. the season) may change in the images (refer to the supplementary for examples). In particular, we evaluate on nine sculptures by the artist Henry Moore. These sets of images contain large variations and the sculpture itself is often smooth, leading to less texture for finding repeatably detectable regions.

We report the results in Table 2 and visualise samples in the supplementary. As can be seen by these results, our approach is able to consistently obtain more 3D points for each image set. Also, in one challenging case D2D succeeds and SIFT fails. These results validate that our dense approach does indeed make our model robust in this context.

4.3 Using Correspondences for Stylization

Previously, we focused on using our matching pipeline for extracting a set of sparse correspondences to be used for localization and 3D reconstruction. Here we evaluate how well our *dense* features can be used for a task that requires dense matching: stylization. The goal is, given two images I_s, I_v of the same scene, to generate an image with the style of I_s but the pose and viewpoint of I_v.

Setup. To achieve this, we first use the D2D model to transform I_s into the position of I_v. The approach is simple: instead of only choosing the mutual nearest neighbours as in (2), we consider

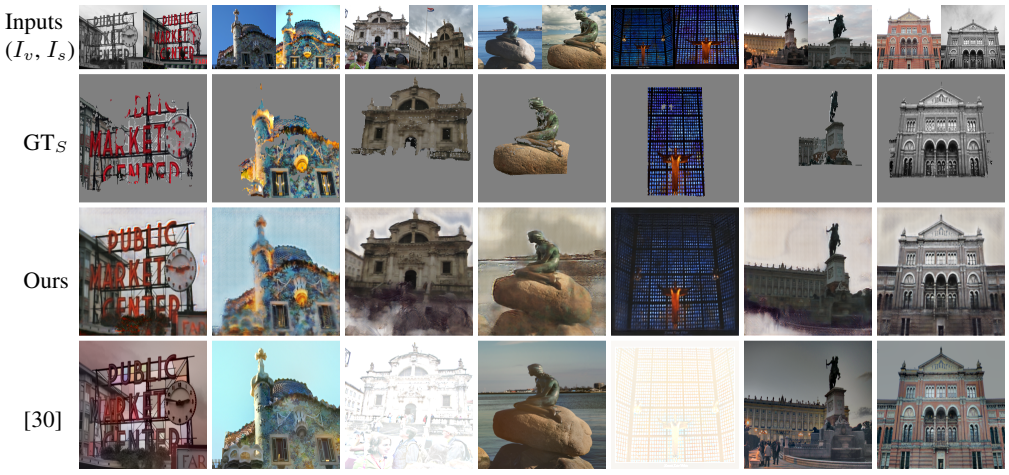


Figure 5: Stylization. Given I_s and I_v, the task is to generate an image with the pose and viewpoint of I_v and style of I_s. We show results for D2D and a baseline that uses semantics [30]. We also show the resampled image (GT$_S$) which is computed using the true correspondences from the MegaDepth dataset [27]. While [30] works well for easy cases, it sometimes copies style from I_v (as shown by the red background in (a), unlit building in (b), and red hued building in (g)). [30] also fails if the semantic prediction is incorrect (e.g. (c) and (e)).

the best match for *every* pixel location. We then use the color of the best match in I_s to color the corresponding location in I_v. This gives us the *sampled image*. The next step is to remove artefacts. We do this by training a refinement model on top of the sampled image in order to obtain an image I_g in the pose of I_v and style of I_s. Full details of the architecture and training are given in the supplementary material.

Comparison to Pix2Pix. In a standard image to image translation task (e.g. Pix2Pix [20]), the two images (e.g. image and semantic map) are aligned. In our case, the images are not aligned. We effectively use our correspondences to align the images and then run a variant of Pix2Pix.

Experimental Setup. To evaluate our results, we use the test set of the MegaDepth dataset (these are landmarks unseen at training time). We randomly select 400 pairs of images and designate one the viewpoint I_v image and the other the style image I_s. We task the models to generate a new image I_g with the style of I_s in the viewpoint of I_v. From the MegaDepth dataset, we can obtain ground truth correspondence for regions in both images and so the true values of I_g for this region. The reported error metric is the mean $L1$ distance between the generated image and true value within this region.

Results. We compare against a stylization approach that uses semantics to perform style transfer [30] in Fig. 5. We also determine the $L1$ error for both setups and obtain 0.22 for [30] and 0.14 for our method, demonstrating that our method is more accurate for regions that can be put in correspondence. The qualitative results demonstrate that our method is more robust, as [30] produces poor results if the semantic prediction is wrong and sometimes copies style from I_v as opposed to I_s (e.g. it creates a colored I_g image when I_s is grey-scale). As we sample from I_s in the first step and then refine the sampled image, our model rarely copies style from I_v. Finally, our full method runs in seconds at test time whereas [30] takes minutes due to a computationally intensive iterative refinement strategy.

5 Conclusion

We introduced a new approach for obtaining correspondences for image pairs. We obtain sota or comparable results on a range of tasks, yet using a smaller network. We achieve these results using a simple strategy of exhaustively comparing features and ranking matches by a distinctiveness score.

Our work has demonstrated that with modern networks, the traditional detect-and-describe approach to remove distracting regions is perhaps no longer necessary. Comparable or better performance can be obtained by a simple exhaustive match over image features. This shows there is the potential for a rethink of SfM pipelines. Instead of using trajectories of points over many images, one could use denser, rougher correspondences between pairs of images. However, we leave this for future work.

Contents

1 Introduction **1**

2 Related Work **2**

3 Method **3**

3.1 D2D Architecture 3

3.2 Training and Loss Functions 4

4 Experiments **5**

4.1 Correspondence Evaluation 5

4.2 Using Correspondences for 3D Reconstruction 6

4.2.1 Camera Localization 6

4.2.2 Structure from Motion (SfM) 7

4.3 Using Correspondences for Stylization 7

5 Conclusion **8**

A Overview **9**

B Additional Experiments **10**

B.1 Further Ablations 10

B.2 Refining Local Matching 10

B.3 SfM Terminology 11

B.4 Further Results for SfM 11

B.5 The YFCC100M Dataset 12

B.6 An InfoNCE Loss 13

B.6.1 Implementation 14

B.6.2 Experiments 14

C Architectures **14**

D Additional Results **15**

D.1 Visualising Confidence Maps 15

D.2 Qualitative Results 16

A Overview

We include additional implementation details and results in Section B. These experiments provide results on an additional dataset, demonstrate how our approach can use a refinement step to achieve state-of-the-art results for all thresholds $>$1px on HPatches, and provide additional ablations including the use of a Noise Contrastive (NCE) loss as opposed to the hinge loss presented in the paper. We define the metrics used in obtaining the SfM results in Section B.3 and visualise our reconstructed 3D models on the large dataset in the video.

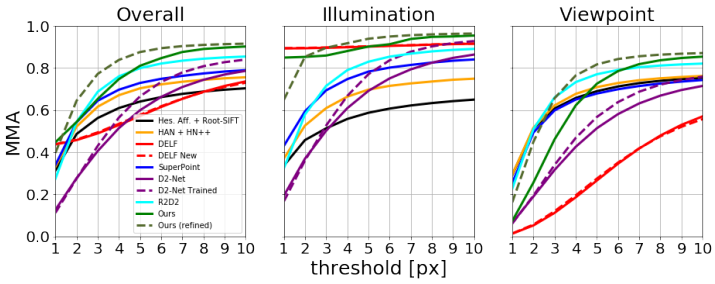


Figure 6: Results on HPatches when refining correspondences. In this experiment, we look at the results of using a refinement strategy based on a local neighbourhood to improve the accuracy of the detected correspondences. We see that using the refinement scheme, we obtain a big boost in performance. In particular, we improve our results for fine-grained pixel thresholds on the viewpoint task. We achieve comparable results with state-of-the-art methods for small pixel thresholds and better performance for larger thresholds ($>$ 4px). On illumination we maintain performance except for very small thresholds ($\leq$ 1px). This small degradation is probably due to limited noise that is introduced with the refinement strategy. In general, using this strategy we improve our results to achieve state of the art performance for all pixel thresholds.

Additionally, we provide further details of the architectures used in Section C for the main model and the stylisation model.

We also provide qualitative samples of the confidence maps learned by our model in Section D.1. We provide additional qualitative results for the stylisation experiment, HPatches dataset, SfM experiment, and Aachen dataset in Section D.2.

B Additional Experiments

In this section we report the results of additional experiments which consider additional ablations of the grid size chosen at test time (Section B.1), a refinement of our model to improve local matching (Section B.2) to achieve state-of-the-art results on HPatches, further comparisons of our model on the 3D reconstruction task (Section B.4), results on the YFCC100M dataset (Section B.5), and results using another popular contrastive loss (NCE) (Section B.6).

B.1 Further Ablations

In this section we discuss and ablate how we select candidate matches at test time.

In order to compare all descriptor vectors at test time, we operate as follows. We create a $G \times G$ pixel grid and bilinearly interpolate on this grid from both the descriptor maps and distinctiveness scores. (Note that we have to normalise the interpolated descriptors.) We consider all pairs of descriptors as candidate matches and compare all descriptors in one image to those in the other. In practice we use $G = 128$.

At test time, we could use a larger grid size for better granularity. However, this comes with the additional computational cost of performing G^4 comparisons. We tried using a larger grid ($G = 256$) on the Aachen-Day Night dataset in Table 5 but obtained comparable results to using $G = 128$. As a result, we continued to use $G = 128$ for all our experiments. Using a larger grid for the larger datasets in SfM (where the number of images are approximately 1500) would have made these experiments intractable using current pipelines.

However, we note that because we only consider points on a grid, we are losing some granularity which presumably impacts the performance on the 3D reconstruction tasks. We discuss a method to refine the correspondences to obtain a finer granularity in the next section and the resulting complications.

B.2 Refining Local Matching

In this section, we demonstrate that our local matching results can be improved by using a simple refinement strategy.

Description of Refinement Strategy. In order to refine the matches obtained using D2D, we use a local neighbourhood to refine the match in the second image. Given a correspondence with location (x_1, y_1) in the first image and (x_2, x_2) in the second image, we look at the similarity score between (x_1, y_1) and the locations in a 3x3 grid centered on (x_2, y_2).

These scores are used to reweight the location in the second image using a normalised weighted sum with one difference. Because the similarity scores are not evenly distribute and cluster around 0.5, they give too much weight to less likely matches. As a result, we subtract the minimum similarity from all scores in a local neighbourhood before computing the normalised weighted sum.

Results. The results are given in Fig. 6. As can be seen, this simple refinement scheme gives a large boost in performance. In particular, using this simple modification gives superior results to state of the art approaches for pixel thresholds $>$ 4px for viewpoint and all thresholds $>$ 2px for illumination. Overall, our model with the refinement scheme achieves state-of-the-art results for all thresholds $>$ 1px.

Discussion. While we could achieve high quality performance on HPatches using this simple modification, we note that it is not straightforward to apply this to the camera localisation or 3D reconstruction tasks. This is because both of these tasks require a single point to be tracked over multiple images in order to perform 3D reconstruction (the camera localisation performs 3D reconstruction as part of the evaluation pipeline). 3D reconstruction pipelines assume a detect and describe pipeline for extracting matches, which implicitly have this assumption baked in to their setup, as they match the same detected points across different images.

However, this assumption is not implicit to our approach, as we only find correspondences between pairs of images at a time. Further refining the points means that the location of a refined point from one pair of images will not necessarily match that of another pair, invalidating the track and negatively affecting the 3D reconstruction. Incorporating these refined points would require rethinking how we incorporate refined correspondences in a 3D reconstruction pipeline. For example, we could use a reference image against which further images are compared and incorporated. Once a reference image has been exhausted, a new reference image would be chosen and so on. We leave such an investigation to future work, but the boost in performance demonstrates the further potential of our dense-to-dense setup.

B.3 SfM Terminology

Here we define the metrics used in reporting the SfM results. Note that these metrics are only *indicative* of the quality of the 3D model; please look at the reconstructed models in the zipped video for a qualitative estimate as to their respective quality.

1. $\uparrow$ # Reg. Ims: **The number of registered images.** This is the number of images that are able to be put into correspondence and for which cameras were obtained when doing the 3D reconstruction. A higher values means more images were registered, implying a better model.
2. $\uparrow$ # Sparse Pts: **The number of sparse points.** This is the number of sparse points obtained after performing the 3D geometry estimation. The higher the number indicates a better model, as more correspondences were able to be triangulated.
3. $\uparrow$ Track Len: **The track length.** How many images a given 3D point is seen in on average. If this is higher, it indicates that the model is more robust, as more images see that 3D point.
4. $\downarrow$ Reproj err: **The repreojection error.** This is the average pixel error between a 3D point and its projection in the images. If this is lower, it indicates the 3D points are more accurate.
5. $\uparrow$ # Dense Points: **The number of dense points.** This is the number of dense points in the final 3D model following the dense matching step. The higher this is, the more of the 3D structure was reconstructed.

B.4 Further Results for SfM

In this section we compare our 3D reconstruction on the Local Feature Evaluation Benchmark [60] to two additional baselines [9, 32] in Table 3. These results were not included in the paper as these

Table 3: SfM. We compare our approach to three baselines on 3D reconstruction for three scenes with a large number of images. Our method obtains superior performance across the metrics except for reprojection error and dense correspondences, despite using coarse correspondences and a single scale. In particular, our method registers more images and obtains more sparse 3D points. ↑ denotes higher is better. ↓ denotes lower is better.

LMark	Method	↑ # Reg. Imgs	↑ # Sparse Pts	↑ Track Len	↓ Reproj. Err	↑ # Dense Pts
Madrid Metropolis 1344 images	RootSIFT [29]	500	116K	6.32	**0.60px**	**1.82M**
	GeoDesc [32]	495	144K	5.97	0.65px	1.56M
	D2 MS [9]	495	144K	**6.39**	1.35px	1.46M
	Ours	**702**	**256K**	6.09	1.30px	1.10M
Gendarmen-markt 1463 images	RootSIFT [29]	1035	338K	5.52	**0.69px**	**4.23M**
	GeoDesc [32]	1004	441K	5.14	0.73px	3.88M
	D2 MS [9]	965	310K	5.55	1.28px	3.15M
	Ours	**1072**	**570K**	**6.60**	1.34px	2.11M
Tower of London 1576 images	RootSIFT [29]	804	239K	**7.76**	**0.61px**	**3.05M**
	GeoDesc [32]	776	341K	6.71	0.63px	2.73M
	D2 MS [9]	708	287K	5.20	1.34px	2.86M
	Ours	**967**	**452K**	5.82	1.32px	1.81M

Table 4: Results on the YFCC dataset [69]. Higher is better. Our approach outperforms all other approaches (e.g. all but RANSAC-Flow) that operate directly on features for smaller angle errors and performs competitively for larger angle errors. Additionally, this dataset clearly demonstrates the utility of our ablations. Note that RANSAC-Flow [62] is a two stage approach that first registers images and then learns optical flow. Such an approach could be added on top of ours.

Method	mAP@5°	mAP@10°
SIFT [29]	46.83	68.03
Contextdesc [31]	47.68	69.55
Superpoint [8]	30.50	50.83
PointCN [38, 76]	47.98	-
PointNet++ [45, 77]	46.23	-
N^3Net [43, 77]	49.13	-
DFE [46, 77]	49.45	-
OANet [77]	52.18	-
RANSAC-Flow [62]	64.68	82.40
Ours (w/o conf)	31.60	40.80
Ours (w/o cond)	53.43	65.13
Ours	55.58	66.79
Ours (E-B1)	**57.23**	**68.39**

additional baselines perform similarly to SIFT [29] and there was limited space. However, we note that both of these baselines use learned descriptors, yet they do not perform any better than SIFT in terms of the number of registered images and sparse 3D points. Our method performs significantly better across all three large scale datasets for obtaining sparse 3D points and registering images, demonstrating the robustness of our approach.

B.5 The YFCC100M Dataset

Here we report results for our model and ablations on the YFCC100M [69] dataset. This dataset further demonstrates the superiority of our approach to other detect and describe setups and the utility of each component of our model (i.e. the confidence and attention mechanism to condition the features on *both* images).

Setup. The task of this dataset is to perform two view geometry estimation on four scenes with 1000 pairs each. Given a pair of images, the task is to use estimated correspondences to predict the essential matrix using the known intrinsic matrices. The essential matrix is decomposed into the rotation and translation component [17]. The reported error metric is the percentage of images that have the rotation and translation error (in degrees) less than a given threshold.

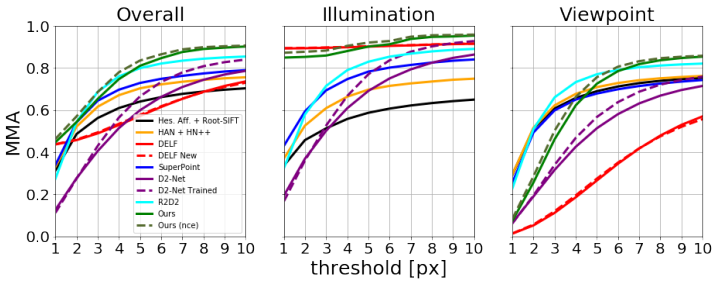


Figure 7: Results on HPatches using a NCE loss. In this experiment, we look at the results of using a NCE loss as opposed to a hinge loss. We see that using a NCE loss, we still achieve high quality, state of the art results, demonstrating the robustness of our approach.

Table 5: Results on Aachen Day-Night using a NCE loss. We can see that training with NCE is slightly worse than our hinge loss, but it is competitive with other state-of-the-art methods. Higher is better. * indicates the method was trained on the Aachen dataset.

Method	Type	Threshold Accuracy		
		0.25m (2°)	0.5m (5°)	5m (10°)
Upright RootSIFT [29]	Spa	36.7	54.1	72.5
DenseSFM [55]	Den	39.8	60.2	84.7
Han+, HN++ [35, 36]	Spa	39.8	61.2	77.6
Superpoint [8]	Spa	42.8	57.1	75.5
DELF [39]	Spa	39.8	61.2	85.7
D2-Net [9]	Spa	**44.9**	66.3	**88.8**
R2D2* [48]	Spa	**45.9**	66.3	**88.8**
Ours (nce)	Den	42.9	62.2	87.8
Ours ($G = 256$)	Den	**44.9**	68.4	87.8
Ours	Den	**44.9**	**70.4**	**88.8**

To run D2D on this dataset, we first use D2D to extract high quality matches for each pair of images. We use the known intrinsics to convert these to points in camera space. We then use RANSAC [11] and the 5-point algorithm in order to obtain the essential matrix [17].

Results. The results are given in Table 4. They demonstrate that our model achieves superior performance for low error thresholds to other methods that directly operate on extracted features. The results further demonstrate that the uniqueness score and conditioning using attention mechanisms are crucial for good performance, validating our model design.

Finally, our model does a bit worse than RANSAC-Flow [62]. However, we note [62] uses a segmentation model to restrict correspondences to only regions in the foreground of the image (e.g. the segmentation model is used to remove correspondences in the sky). Additionally, this method first registers images under a homography using predetected correspondences and then trains a network on top of the transformed images to perform fine-grained optical flow. As a result, this method performs as well as the underlying correspondences. Considering that our method has consistently been demonstrated to perform comparably or better than previous approaches for obtaining correspondences, we note that this method could be used on top of ours for presumably further improved performance. However, as this unpublished work was only recently released on arXiv, we leave this for future work.

B.6 An InfoNCE Loss

In the paper we demonstrate the robustness of our approach to the precise choice of architecture (e.g. we can achieve impressive results using a ResNet [18] or EfficientNet [68] backbone).

Here, we consider using the CPC objective [42], which is inspired by Noise-Contrastive Estimation (NCE) approaches [14, 37], and so is called an InfoNCE loss. While this is also a contrastive loss, similarly to the hinge loss in the main paper, the implementation is different. We find that we can still

achieve impressive results with this loss, demonstrating the robustness of the approach to the precise choice of loss.

B.6.1 Implementation

We follow the implementation of [42], except that because we use normalised features, we add a temperature τ. This is essential for good performance.

The setup is the same as that described in the paper, except for the implementation of the loss. Assume we have two descriptor maps D^1 and D^1 corresponding to the two input images I^1 and I^2. At a location i in D^1, we obtain the descriptor vector $d_i^1 \in \mathbb{R}^c$. To compare descriptor vectors, we first normalise and then use the scalar product to obtain a scalar matching score:

$$s(d_i^1, d_j^2) = \left(\frac{d_i^1}{||d_i^1||_2} \right)^T \frac{d_j^2}{||d_j^2||_2}. \tag{6}$$

If the score is near 1, this is most likely a match. If it is near -1, it is most likely not a match.

Again, as in the paper, given two images of a scene I^1 and I^2 with a known set of correspondences from MegaDepth [27], we randomly select a set $\mathcal{P}$ of L true correspondences. For each positive correspondence p, we additionally select a set $\mathcal{N}_p$ of K negative correspondences. The loss is then

$$\mathcal{L}_{\texttt{nce}} = -\log \frac{1}{L} \sum_{p=(x,y)\in\mathcal{P}} \frac{\texttt{exp}(\tau * s(d_x^1, d_y^2)}{\texttt{exp}(\tau * s(d_x^1, d_y^2)) + \sum_{(x,\hat{y})\in\mathcal{N}_p} \texttt{exp}(\tau * s(d_x^1, d_{\hat{y}}^2))} \tag{7}$$

where $\tau = 20$ is a temperature.

B.6.2 Experiments

We train the InfoNCE model using the $\mathcal{L}_{\texttt{nce}}$ in the same manner as in the paper and evaluate it on two of the datasets discussed in the paper: HPatches [2] and Aachen [55, 56].

HPatches. The results are given in Fig. 7. From here we see that our model with an NCE loss performs competitively on this dataset, obtaining superior results to that of the model in the paper.

Aachen Day-Night. The results are given in Table 5. These results demonstrate that using an NCE loss with our backbone achieves results competitive with other state-of-the-art approaches but it performs a bit worse than the hinge loss used in the paper.

Discussion. These experiments have shown that we can achieve high quality results when using a different but effective contrastive loss. As a result, our approach is robust to not only the backbone architecture (as shown in the paper) but also the precise choice of the contrastive loss.

C Architectures

Architecture for Main Model. The components of the main model are described in the main text. Here we give further details of the different components. The encoder is a ResNet50 model, except that we extract the features from the last two blocks to obtain feature maps f_S^i and f_L^i. The details are given in Table 6.

The features f_L^i and f_S^i are projected in order to reduce the number of channels using linear layers. There are four linear layers (one for each of f_L^1, f_L^2, f_S^1, f_S^2). The linear layers operating at the larger resolution (f_L^i) project the features from 2048 size vectors to 256. The linear layers operating at the smaller resolution (f_S^i) project the features from 1024 size vectors to 128.

The decoder consists of a sequence of decoder layers. A layer takes the bi-linearly upsampled features from the previous layer, the corresponding encoded features, and optionally the attended features. The details are given in Table 7. Finally, the unnormalised features are passed to a MLP which regresses the distinctiveness score. The MLP consists of three blocks of linear layer (with no bias) and batch normalisation followed by a sigmoid layer. The channel dimensions are $64 \to 1 \to 1$.

Architecture for Stylisation. In Fig. 8, we illustrate further our method for stylising images using our initial set of dense correspondences. Given two images I_v and I_S, the task is to generate an

Table 6: Encoder of Main Model. The encoder is a Resnet50 [18] encoder. The convolutions column denotes the convolutional and max-pooling operations. Implicit are the BatchNorm and ReLU operations that follow each convolution.

layer name	output size	convolutions
conv 1	128×128	$7 \times 7, 64$, stride 2
conv2_x	64×64	3×3 max pool, stride 2 $\begin{pmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{pmatrix} \times 3$
conv3_x	32×32	$\begin{pmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{pmatrix} \times 4$
conv4_x (f_L^i)	16×16	$\begin{pmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{pmatrix} \times 6$
conv5_x (f_S^i)	8×8	$\begin{pmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{pmatrix} \times 3$

Table 7: Decoder of Main Model. The decoder is a UNet [53] variant. The convolutions column denotes the convolutional operations. Implicit are the BatchNorm and ReLU operations that follow each convolution as well as the bi-linear upsampling operation that resizes features from the previous layer before the convolutional blocks.

layer name	inputs	output size	convolutions
deconv_5	conf5_x ($\times 2$), $\hat{f}_S^i$	16×16	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix}$
deconv_4	deconv_5, conv4_x, $\hat{f}_L^i$	32×32	$\begin{pmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{pmatrix}$
deconv_3	deconv_4, conv3_x	64×64	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix}$
deconv_2	deconv_3, conv2_x	128×128	$\begin{pmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{pmatrix}$
deconv_1	deconv_2, conv1_x	256×256	$\begin{pmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{pmatrix}$

image with the viewpoint of I_v and the style of I_S. In brief, we first use the dense correspondences to sample from I_S to obtain the initial image. We then use a refinement network to fix errors and fill in missing regions. We do this in two stages. The first stage fixes errors and can be trained with an L1 loss. However, we wish to use a discriminator loss, but using this directly on the intermediary image causes information to leak and the generated image to match I_v, which is input to the network. To use a discriminator loss without leaking information, we use a second network (a sequence of ResNet blocks) which is trained with both an L1 and discriminator loss. Crucially, we do not allow gradients to flow from the second network (the set of ResNet blocks) to the first.

D Additional Results

D.1 Visualising Confidence Maps

In Fig. 9 we illustrate the predicted confidence maps. Here we can see that the most confident parts of both images are regions that exist in *both* images. We further test this by looking at what the confident maps look like when we keep one input view the same but change the other in Fig. 10. We can see that the confidence maps change depending on the input images: the output is indeed dependent on *both* input images.

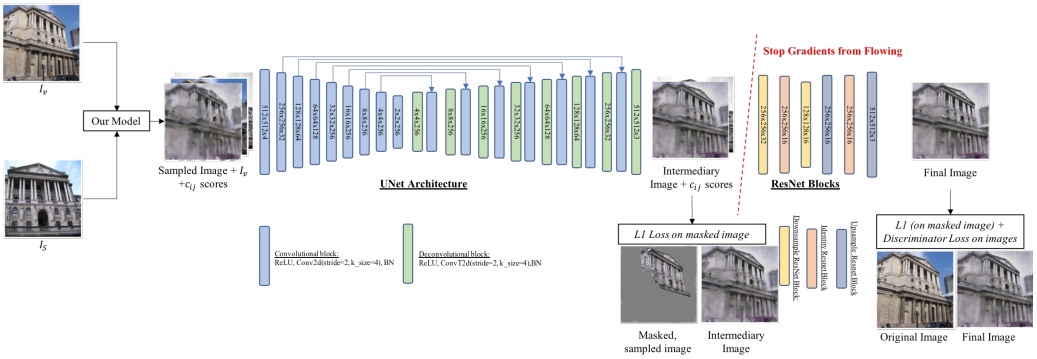


Figure 8: Illustration of the architecture used for the downstream stylization task. We first use our model to predict correspondences to transform the input image I_S into the position of I_v. The next step is to learn how to fill in and fix errors with a discriminator. However, we additionally can train the portion of the generated image visible in both input images to match the true transformed image. We also can use high frequency information in I_v when performing this transformation. However, if we train end-to-end then information will leak from I_v to the generated image. As a result, we train in two stages. We first generate an intermediary image using a UNet [53] which is trained using an L1 loss on the generated intermediary image for regions that are in common between the two images (and for which we can determine what the true pixel colour should be). We input this intermediary image to a set of ResNet blocks to refine the original prediction: this is trained with both a discriminator (pix2pixHD [72]) and L1 loss.



Figure 9: Random pairs associated with their predicted confidence on the MegaDepth test set. Top row shows the input pairs, bottom row the associated confidence maps.

D.2 Qualitative Results

In Fig. 11 we show random matches obtained by our method on random samples from the Aachen Day-Night test set and similarly in Fig. 12 for HPatches, Fig. 13 for 3D reconstruction using SfM and Fig. 14 for the stylisation task.



Figure 10: Random pairs associated with their predicted confidence on the MegaDepth test set. Top: Pairs of image of the same scene. Bottom: Pairs with one image from the top associated with another from a different scene. Notice that the confidence changes between the two cases and becomes irrelevant.

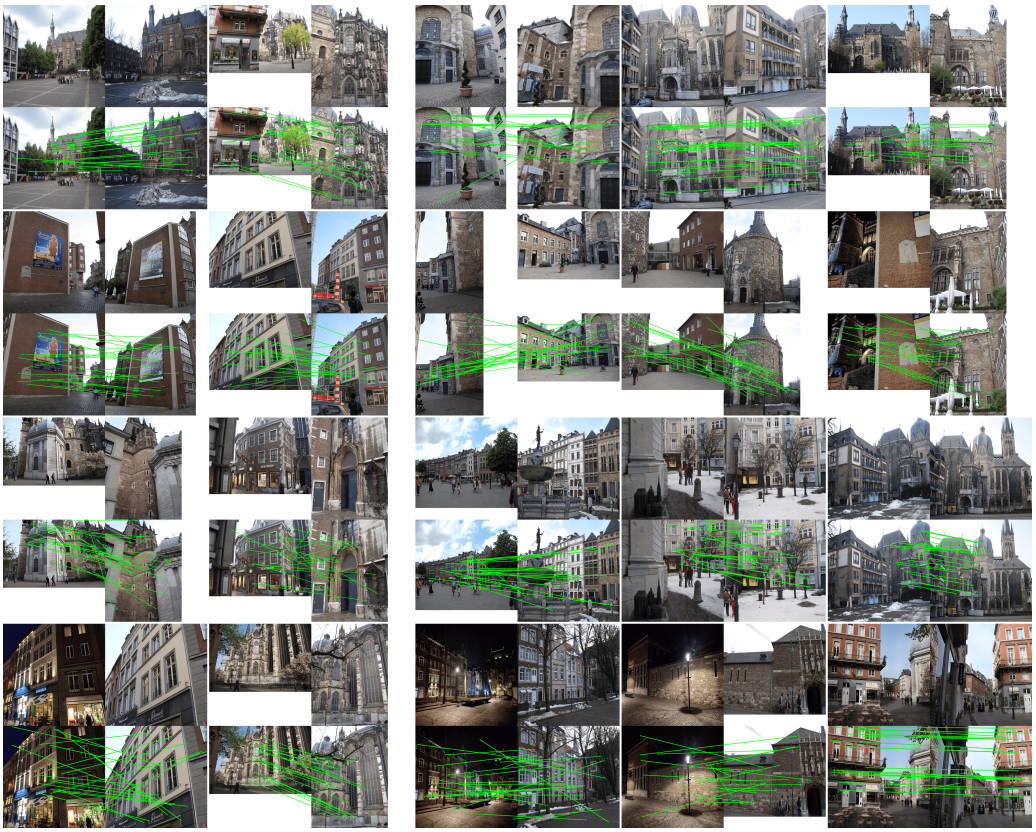


Figure 11: Random sample matches found on the Aachen Day-Night test set. We show the original image pairs top and the pairs with a random subset of located correspondences overlaid below.

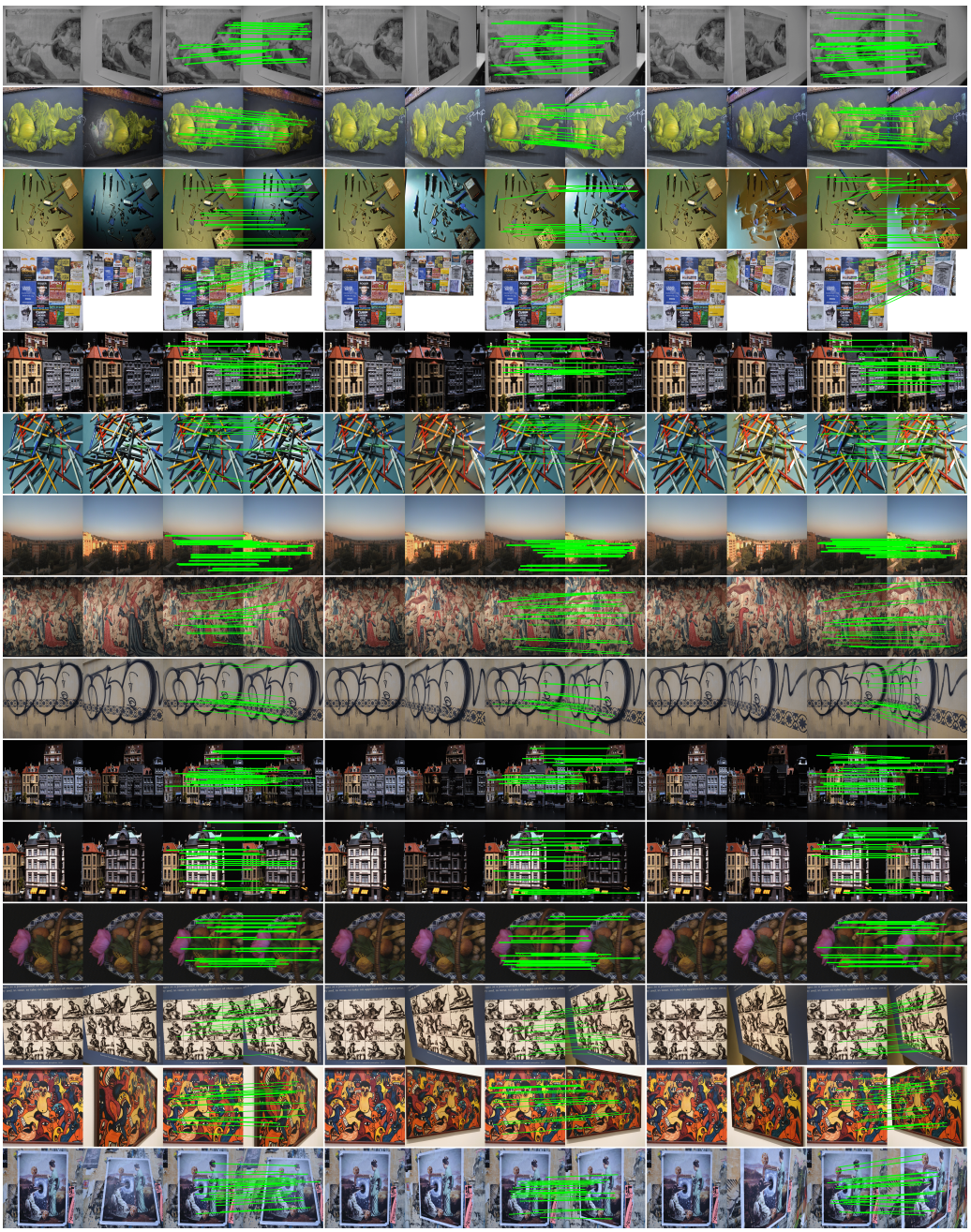


Figure 12: Random sample matches found on the HPatches test set. Rows show progressively harder illumination or viewpoint changes. We first show the original image pairs followed by the pairs with a random subset of located correspondences overlaid.



Input Images | Our 3D Reconstruction | SIFT [29] 3D Reconstruction

Figure 13: Additional SfM results. Randomly selected input images and 3D models reconstructed using our matches and those obtained using the SIFT [29] baseline. This figure demonstrates the variety of the input images and their *scene shift* as well as that both our and [29] are of similar quality for these image sets.

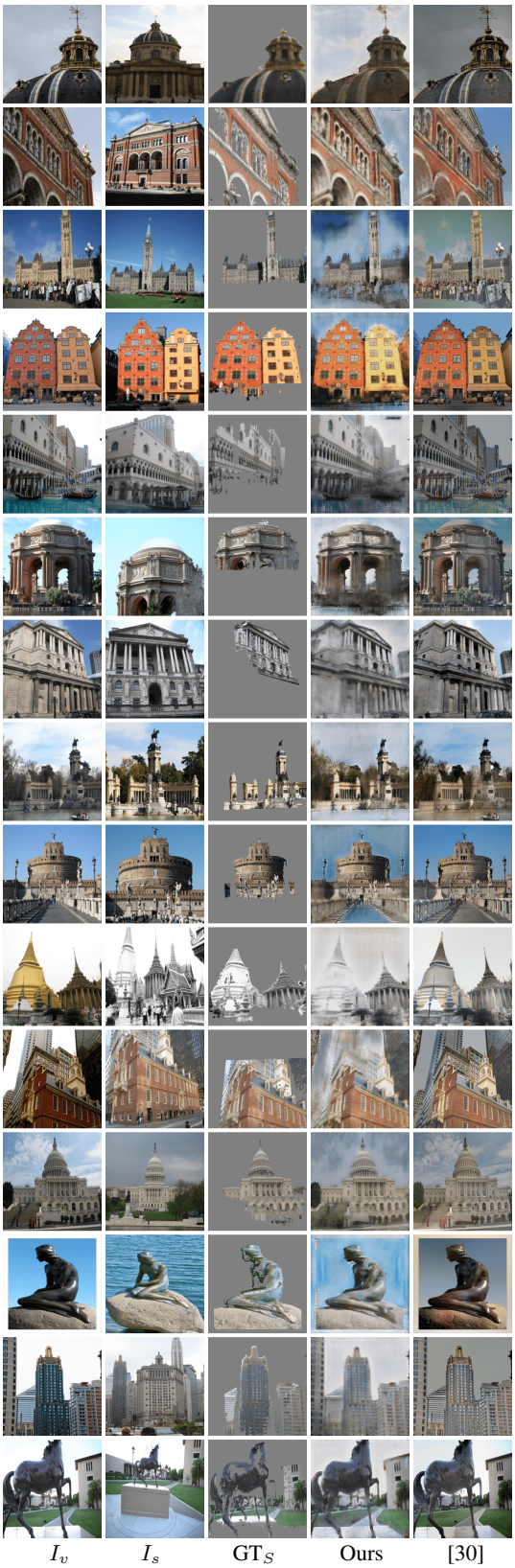


Figure 14: Additional stylisation results. These results are randomly sampled from the test set of MegaDepth.

References

[1] R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *Proc. CVPR*, 2012.

[2] V. Balntas, K. Lenc, A. Vedaldi, and K. Mikolajczyk. Hpatch: A benchmark and evaluation of handcrafted and learned local descriptors. In *Proc. CVPR*, 2017. V. Balntas and K. Lenc contributed equally to this work.

[3] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proc. BMVC*, 2016.

[4] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. PatchMatch: A randomized correspondence algorithm for structural image editing. In *ACM Transactions on Graphics (ToG)*, volume 28, page 24. ACM, 2009.

[5] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *Proc. ECCV*, May 2006.

[6] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *Proc. ECCV*, 2010.

[7] C. B. Choy, J. Gwak, S. Savarese, and M. Chandraker. Universal correspondence network. In *NeurIPS*, 2016.

[8] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR workshops*, 2018.

[9] M. Dusmanu, I. Rocco, T. Pajdla, M. Pollefeys, J. Sivic, A. Torii, and T. Sattler. D2-Net: A trainable cnn for joint description and detection of local features. In *Proc. CVPR*, 2019.

[10] M. E. Fathy, Q.-H. Tran, M. Zeeshan Zia, P. Vernaza, and M. Chandraker. Hierarchical metric learning and matching for 2D and 3D geometric correspondences. In *Proc. ECCV*, 2018.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM*, 24(6):381–395, 1981.

[12] D. F. Fouhey, A. Gupta, and A. Zisserman. 3d shape attributes. In *Proc. CVPR*, 2016.

[13] H. Germain, G. Bourmaud, and V. Lepetit. S2DNet: Learning accurate correspondences for sparse-to-dense feature matching. *arXiv preprint arXiv:2004.01673*, 2020.

[14] M. Gutmann and A. Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proc. AISTATS*, 2010.

[15] M. J. Hannah. *Computer matching of areas in stereo images.* PhD thesis, 1974.

[16] C. G. Harris and M. Stephens. A combined corner and edge detector. In *Proc. Alvey Vision Conf.*, pages 147–151, 1988.

[17] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

[18] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[19] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. CVPR*, 2017.

[20] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*, 2017.

[21] Y. Ke and R. Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proc. CVPR*, Jun 2004.

[22] S. Kim, S. Lin, S. R. Jeon, D. Min, and K. Sohn. Recurrent transformer networks for semantic correspondence. In *NeurIPS*, 2018.

[23] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014.

[24] Z. Lai, E. Lu, and W. Xie. MAST: A memory-augmented self-supervised tracker. In *Proc. CVPR*, 2020.

[25] S. Leutenegger, M. Chli, and R. Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proc. ICCV*, pages 2548–2555, 2011.

[26] S. Li, K. Han, T. W. Costain, H. Howard-Jenkins, and V. Prisacariu. Correspondence networks with adaptive neighbourhood consensus. In *Proc. CVPR*, 2020.

[27] Z. Li and N. Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proc. CVPR*, 2018.

[28] C. Liu, J. Yuen, and A. Torralba. Sift Flow: Dense correspondence across scenes and its applications. *IEEE PAMI*, 33(5):978–994, 2010.

[29] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.

[30] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. *Proc. CVPR*, 2017.

[31] Z. Luo, T. Shen, L. Zhou, J. Zhang, Y. Yao, S. Li, T. Fang, and L. Quan. ContextDesc: Local descriptor augmentation with cross-modality context. In *Proc. CVPR*, 2019.

[32] Z. Luo, T. Shen, L. Zhou, S. Zhu, R. Zhang, Y. Yao, T. Fang, and L. Quan. GeoDesc: Learning local descriptors by integrating geometry constraints. In *Proc. ECCV*, 2018.

[33] K. Mikolajczyk and C. Schmid. Scale & affine invariant interest point detectors. *IJCV*, 1(60):63–86, 2004.

[34] K. Mikolajczyk, T. Tuytelaars, C. Schmid, A. Zisserman, J. Matas, F. Schaffalitzky, T. Kadir, and L. Van Gool. A comparison of affine region detectors. *IJCV*, 65(1/2):43–72, 2005.

[35] A. Mishchuk, D. Mishkin, F. Radenovic, and J. Matas. Working hard to know your neighbor's margins: Local descriptor learning loss. In *NeurIPS*, 2017.

[36] D. Mishkin, F. Radenovic, and J. Matas. Repeatability is not enough: Learning affine regions via discriminability. In *Proc. ECCV*, 2018.

[37] A. Mnih and K. Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. In *NeurIPS*, 2013.

[38] K. Moo Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to find good correspondences. In *Proc. CVPR*, 2018.

[39] H. Noh, A. Araujo, J. Sim, T. Weyand, and B. Han. Large-scale image retrieval with attentive deep local features. In *Proc. ICCV*, 2017.

[40] D. Novotny, D. Larlus, and A. Vedaldi. Anchornet: A weakly supervised network to learn geometry-sensitive features for semantic matching. In *Proc. CVPR*, 2017.

[41] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. LF-Net: learning local features from images. In *NeurIPS*, 2018.

[42] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[43] T. Plötz and S. Roth. Neural nearest neighbors networks. In *NeurIPS*, 2018.

[44] M. Pollefeys. *Self-calibration and metric 3D reconstruction from uncalibrated image sequences*. PhD thesis, ESAT-PSI, K.U.Leuven, 1999.

[45] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017.

[46] R. Ranftl and V. Koltun. Deep fundamental matrix estimation. In *Proc. ECCV*, 2018.

[47] A. Ranjan and M. J. Black. Optical flow estimation using a spatial pyramid network. In *Proc. CVPR*, 2017.

[48] J. Revaud, P. Weinzaepfel, C. De Souza, N. Pion, G. Csurka, Y. Cabon, and M. Humenberger. R2D2: Repeatable and reliable detector and descriptor. In *NeurIPS*, 2019.

[49] I. Rocco, R. Arandjelovic, and J. Sivic. Convolutional neural network architecture for geometric matching. In *Proc. CVPR*, 2017.

[50] I. Rocco, R. Arandjelovic, and J. Sivic. End-to-end weakly-supervised semantic alignment. In *Proc. CVPR*, 2017.

[51] I. Rocco, R. Arandjelović, and J. Sivic. Efficient neighbourhood consensus networks via submanifold sparse convolutions. *CoRR*, abs/2004.10566, 2020.

[52] I. Rocco, M. Cimpoi, R. Arandjelović, A. Torii, T. Pajdla, and J. Sivic. Neighbourhood consensus networks. In *NeurIPS*, 2017.

[53] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Springer, 2015.

[54] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski. Orb: An efficient alternative to SIFT or SURF. In *Proc. ICCV*, pages 2564–2571, 2011.

[55] T. Sattler, W. Maddern, C. Toft, A. Torii, L. Hammarstrand, E. Stenborg, D. Safari, M. Okutomi, M. Pollefeys, J. Sivic, F. K. Kahl, and T. Pajdla. Benchmarking 6dof outdoor visual localization in changing conditions. In *Proc. CVPR*, 2018.

[56] T. Sattler, T. Weyand, B. Leibe, and L. Kobbelt. Image retrieval for image-based localization revisited. In *Proc. BMVC*, 2012.

[57] N. Savinov, L. Ladicky, and M. Pollefeys. Matching neural paths: transfer from recognition to correspondence search. In *NeurIPS*, 2017.

[58] N. Savinov, A. Seki, L. Ladicky, T. Sattler, and M. Pollefeys. Quad-Networks: unsupervised learning to rank for interest point detection. In *Proc. CVPR*, 2017.

[59] J. L. Schönberger and J.-M. Frahm. Structure-from-motion revisited. In *Proc. CVPR*, 2016.

[60] J. L. Schönberger, H. Hardmeier, T. Sattler, and M. Pollefeys. Comparative evaluation of hand-crafted and learned local features. In *Proc. CVPR*, 2017.

[61] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proc. ECCV*, 2016.

[62] X. Shen, F. Darmon, A. A. Efros, and M. Aubry. RANSAC-Flow: generic two-stage image alignment. *arXiv preprint arXiv:2004.01526*, 2020.

[63] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proc. ICCV*, 2015.

[64] K. Simonyan, A. Vedaldi, and A. Zisserman. Learning local feature descriptors using convex optimisation. *IEEE PAMI*, 2014.

[65] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz. PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In *Proc. CVPR*, 2018.

[66] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum. Symmetric stereo matching for occlusion handling. In *Proc. CVPR*, 2005.

[67] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[68] M. Tan and Q. V. Le. EfficientNet: Rethinking model scaling for convolutional neural networks. *Proc. ICML*, 2019.

[69] B. Thomee, D. A. Shamma, G. Friedland, B. Elizalde, K. Ni, D. Poland, D. Borth, and L.-J. Li. Yfcc100m: the new data in multimedia research. *Communications of the ACM*, 59(2):64–73, 2016.

[70] E. Tola, V. Lepetit, and P. Fua. A fast local descriptor for dense matching. In *Proc. CVPR*, 2008.

[71] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *Proc. ECCV*, 2018.

[72] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proc. CVPR*, 2018.

[73] X. Wang, A. Jabri, and A. A. Efros. Learning correspondence from the cycle-consistency of time. In *Proc. CVPR*, 2019.

[74] O. J. Woodford, P. H. S. Torr, I. D. Reid, and A. W. Fitzgibbon. Global stereo reconstruction under second order smoothness priors. *IEEE PAMI*, 31(12):2115–2128, 2009.

[75] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *Proc. ECCV*, 2016.

[76] J. Zhang, D. Sun, Z. Luo, A. Yao, L. Zhou, T. Shen, Y. Chen, L. Quan, and H. Liao. Learning two-view correspondences and geometry using order-aware network. In *Proc. ICCV*, 2019.

[77] L. Zhang and S. Rusinkiewicz. Learning to detect features in texture images. In *Proc. CVPR*, 2018.

Chapter 4

Learning to Predict 3D Surfaces of Sculptures from Single and Multiple views.

This work was published in IJCV in 2019. It is an extended version of our 2017 BMVC paper that won *Best Poster*.

This work introduces a challenging dataset of 3D sculptures for evaluating models that predict 3D shape. This chapter also introduces a model for representing shapes as an embedding, such that multiple views can be combined at the embedding level to improve the 3D prediction. Images can be meaningfully combined, *even* if the images are not photometrically consistent, as introduced in Section 1.2.

IJCV manuscript No.
(will be inserted by the editor)

Learning to Predict 3D Surfaces of Sculptures from Single and Multiple Views

Olivia Wiles · Andrew Zisserman

Received: date / Accepted: date

Abstract The objective of this work is to reconstruct the 3D surfaces of sculptures from one or more images using a view-dependent representation. To this end, we train a network, `SiDeNet`, to predict the **Si**lhouette and **De**pth of the surface given a variable number of images; the silhouette is predicted at a different viewpoint from the inputs (e.g. from the *side*), while the depth is predicted at the viewpoint of the input images. This has three benefits. First, the network learns a representation of shape beyond that of a single viewpoint, as the silhouette forces it to respect the visual hull, and the depth image forces it to predict concavities (which don't appear on the visual hull). Second, as the network learns about 3D using the proxy tasks of predicting depth and silhouette *images*, it is not limited by the resolution of the 3D representation. Finally, using a view-dependent representation (e.g. additionally encoding the viewpoint with the input image) improves the network's generalisability to unseen objects.

Additionally, the network is able to handle the input views in a flexible manner. First, it can ingest a different number of views during training and testing, and it is shown that the reconstruction performance improves as additional views are added at test-time. Second, the additional views do not need to be photometrically consistent.

The network is trained and evaluated on two synthetic datasets – a realistic sculpture dataset (*SketchFab*), and ShapeNet. The design of the network is validated by comparing to state of the art methods for a set of tasks. It is shown that (i) passing the input viewpoint (i.e. using a view-dependent representation) improves the network's generalisability at test time. (ii) Predicting depth/silhouette images allows for higher quality predictions in 2D, as the network is not limited by the chosen latent 3D representation. (iii) On both datasets the method of combining views in a global manner performs better than a local method.

Finally, we show that the trained network generalizes to real images, and probe how the network has encoded the latent 3D shape.

Keywords Visual hull · Generative model · Silhouette prediction · Depth prediction · Convolutional neural networks · Sculpture dataset

O. Wiles
Department of Engineering Science, University of Oxford
E-mail: ow@robots.ox.ac.uk

A. Zisserman
Department of Engineering Science, University of Oxford

1 Introduction

Learning to infer the 3D shape of complex objects given only a few images is one of the grand challenges of computer vision. Another of the many benefits of deep learning has been a resurgence of interest in this task. Many recent works have developed the idea of inferring 3D shape given a set of classes (e.g. cars, chairs, rooms). This modern treatment of class based reconstruction follows on from the pre-deep learning classic work of Blanz and Vetter (1999) for faces and later for other classes such as semantic categories (Kar et al (2015); Cashman and Fitzgibbon (2013)) or cuboidal room structures (Fouhey et al (2015); Hedau et al (2009)).

This work extends this area in two directions: first, it considers 3D shape inference from *multiple* images rather than a single one (though this is considered as

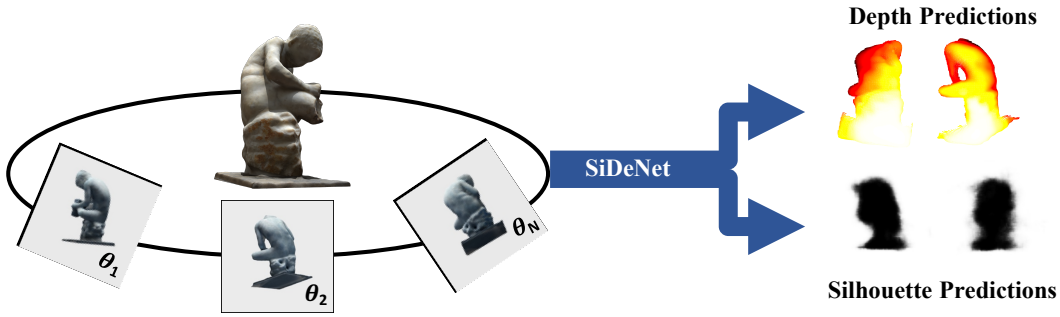


Fig. 1 An overview of SiDeNet. First, images of an object are taken at various viewpoints $\theta_1 ... \theta_N$ by rotating the object about the vertical axis. Given a set of these views (the number of which may vary at test time), SiDeNet predicts the depth of the sculpture at the given views and the silhouette at a new view θ'. Here, renderings of the predicted depth at two of the given views and silhouette predictions at new viewpoints are visualised. The depth predictions are rendered using the depth value for the colour (e.g. dark red is further away and yellow/white nearer).

well); second, it considers the quite generic class of piecewise smooth textured sculptures and the associated challenges.

To achieve this, a deep learning architecture is introduced which can take into account a variable number of views in order to predict depth for the given views and the silhouette at a new view (see figure 1 for an overview). This approach has a number of benefits: *first* the network learns how to combine the given views – it is an architectural solution – without using multi view stereo. As a result, the views need not be photometrically consistent. This is useful if the views exhibit changes in exposure/lighting/texture or are taken in different contexts (so one may be damaged), etc. By enforcing that the same network must be able to predict 3D from single and multiple views, the network must be able to infer 3D shape using global information from one view and combine this information given multiple views; this is a different approach from building up depth locally using correspondences as would be done in a traditional multi view stereo approach.

Second, using a view-dependent representation means that the model makes few assumptions about the distribution of input shapes or their orientation. This is especially beneficial if there is no canonical frame or natural orientation over the input objects (e.g. a chair facing front and upright is at 0°). This generalisation power is demonstrated by training/evaluating SiDeNet on a dataset of sculptures which have a wide variety of shapes and textures. SiDeNet generalises to new unseen shapes without requiring any changes.

Finally, as only image representations are used, the quality of the 3D model is not limited by the 3D resolution of a voxel grid or a finite set of points but by the image resolution.

Contributions. This work brings the following contributions. *First*, a fully convolutional architecture and loss function, termed *SiDeNet* (sections 3,4) is introduced for understanding 3D shape. It can incorporate additional views at test time, and the predictions improve as additional views are incorporated when both using 2D convolutions to predict depth/silhouettes as well as 3D convolutions to latently infer the 3D shape. Further, this is true without assuming that the objects have a canonical representation unlike many contemporary methods. *Second*, a dataset of complex sculptures which are augmented in 3D (section 5). This dataset demonstrates that the learned 3D representation is sufficient for silhouette prediction as well as new view synthesis for a set of unseen objects with complex shapes and textures. *Third*, a thorough evaluation that demonstrates how incorporating additional views improves results and the benefits of the data augmentation scheme (section 6) as well as that SiDeNet can be used directly on real images. This evaluation also demonstrates how SiDeNet can incorporate multiple views *without* requiring photometric consistency and demonstrates that SiDeNet is competitive or better than comparable state-of-the-art methods for 3D prediction and at leveraging multiple views on both the *Sculptures* and *ShapeNet* datasets. *Finally*, the architecture is investigated to determine how information is encoded and aggregated across views in section 8.[1]

This work is an extension of that described in Wiles and Zisserman (2017). The original architecture is referred to as SilNet, and the improved architecture (the subject of this work) SiDeNet. SilNet learns about the visual hull of the object and is trained on images of a small resolution size to predict the silhouette of the object at again a small resolution size. This is improved in this work, SiDeNet. The loss function is improved by adding an additional term for depth that enforces

[1] Data and resources are available at `http://www.robots.ox.ac.uk/~vgg/data/SilNet/`.

that the network should learn to predict concavities on the 3D shape (section 3). The architecture is improved by increasing the resolution of the input and predicted image (section 4). The dataset acquisition phase is improved by adding data augmentation in 3D (section 5). These changes are analysed in section 6.

2 Related work

Inferring 3D shape from one or more images has a long history in computer vision. However, single vs multi-image approaches have largely taken divergent routes. Multi-image approaches typically enforce geometric constraints such that the estimated model satisfies the silhouette and photometric constraints imposed by the given views whereas single image approaches typically impose priors in order to constrain the problem. However, recent deep learning approaches have started to tackle these problems within the same model. This section is divided into three areas: multi-image approaches and single image approaches without deep learning, and newer deep learning approaches which attempt to combine these two problems into one model.

2.1 Multi-image

Traditionally, given multiple images of an object, 3D can be estimated by tracking feature points across multiple views; these constraints are then used to infer the 3D at the feature points using structure-from-motion (SfM), as explained in Hartley and Zisserman (2004). Additional photometric and silhouette constraints can also be imposed on the estimated shape of the object. Silhouette based approaches that attempt to learn the visual hull (introduced by Laurentini (1994)) using a set of silhouettes with known camera positions can be done in 3D using voxels (or another 3D representation) or in the image domain by interpolating between views (e.g. the work of Matusik et al (2000)). This is improved by other approaches which attempt to construct the latent shape subject to the silhouette as well as photometric constraints; they differ in how they represent the shape and how they enforce the geometric and photometric constraints (Boyer and Franco (2003); Kolev et al (2009); Vogiatzis et al (2003) – see Seitz et al (2006) for a thorough review). The limitation of these approaches is that they require multiple views of the object at test time in order to impose constraints on the generated shape and they cannot extrapolate to unseen portions of the object.

2.2 Single image

When given a single image, then correspondences cannot be used to derive the 3D shape of the model. As a result, single-image approaches must impose priors in order to recover 3D information. The prior may be based on the class by modelling the deviation from a mean shape. This approach was introduced in the seminal work of Blanz and Vetter (1999). The class based reconstruction approach has continued to be developed for semantic categories (Cashman and Fitzgibbon (2013); Prasad et al (2010); Vicente et al (2014); Xiang et al (2014); Kar et al (2015); Rock et al (2015); Kong et al (2017)) or cuboidal room structures (Fouhey et al (2015); Hedau et al (2009)). Another direction is to use priors on shading, texture, or illumination to infer aspects of 3D shape (Zhang et al (1999); Blake and Marinos (1990); Barron and Malik (2015); Witkin (1981)).

2.3 Deep learning approaches

Newer deep learning approaches have traditionally built on the single image philosophy of learning a prior distribution of shapes for a given object class. However, in these cases the distribution is implicitly learned for a specific object class from a single image using a neural network. These methods rely on a large number of images of a given object class that are usually synthetic. The distribution may be learned by predicting the corresponding 3D shape from a given image for a given object class using a voxel, point cloud, or surface representation (Girdhar et al (2016); Wu et al (2016); Fan et al (2016); Sinha et al (2017); Yan et al (2016); Tulsiani et al (2017); Rezende et al (2016); Wu et al (2017)). These methods differ in whether they are supervised or use a weak-supervision (e.g. the silhouette or photometric consistency as in Yan et al (2016); Tulsiani et al (2017)). A second set of methods learn a latent representation by attempting to generate new views conditioned on a given view. This approach was introduced in the seminal work of Tatarchenko et al (2016) and improved on by Zhou et al (2016); Park et al (2017).

While demonstrating impressive results, these deep learning methods methods are trained/evaluated on a single or small number of object classes and often do not consider the additional benefits of multiple views. The following approaches consider how to generalise to multiple views and/or the real domain.

The approaches that consider the multi-view case are the following. Choy et al (2016) use a recurrent neural network on the predicted voxels given a sequence of images to reconstruct the model. Kar et al (2017) use

the known camera position to impose geometric constraints on how the views are combined in the voxel representation. Finally, Soltani et al (2017) pre-determine a fixed set of viewpoints of the object and then train a network for silhouette/depth from these known viewpoints. However, changing any of the input viewpoints or output viewpoints would require training a new network.

More recent approaches such as the works of Zhu et al (2017); Wu et al (2017) have attempted to finetune the model trained on synthetic data on real images using the silhouette or another constraint, but they only extend to semantic classes that have been seen in the synthetic data. Novotny et al (2017) directly learn on real data using 3D reconstructions generated by a SfM pipeline. However, they require many views of the same object and enough correspondences at train time in order to make use of the SfM pipeline.

This paper improves on previous work in three ways. *First* an image based approach is used for predicting the silhouette and depth, thereby enforcing that the latent model learns about 3D shape without having to explicitly model the full 3D shape. *Second* our method of combining multiple views using a latent embedding acts globally as opposed to locally (e.g. Choy et al (2016) combine information for subsets of voxels and Kar et al (2017) combine information along projection rays). Additionally, our method does not require photometric consistency or geometric modelling of the camera movement and intrinsic parameters – it is an architectural solution. In spirit, our method of combining multiple views is more similar to multi-view classification/recognition architectures such as the works of Su et al (2015); Qi et al (2016). *Third* a new *Sculptures* dataset is curated from SketchFab (2017) which exhibits a wide variety of shapes from many semantic classes. Many contemporary methods train/test on ShapeNet core which contains a set of semantic classes. Training on class-specific datasets raises the question: to what extent have these architectures actually learnt about shape and how well will they generalise to unseen objects that vary widely from the given class (e.g. as an extreme how accurately would these models reconstruct a tree when trained on beds/bookcases). We investigate this on the *Sculptures* dataset.

3 Silhouette & depth: a multi-task loss

The loss function used enforces two principles: first that the network learns about the visual hull, and second that it learns to predict the surface (and thus also concavities) at the given view. This is done by predicting, for a given image (or set of images), the silhouette in a new view and the depth at the given views. We expand on these two points in the following.

3.1 Silhouette

The first task considered is how to predict the silhouette at a new view given a set of views of an object. The network can do well at this task only if it has learned about the 3D shape of the object. To predict the silhouette at new angle θ', the network must at least encode the visual hull (the visual hull is the volume swept out by the intersection of the back-projected silhouettes of an object as the viewpoint varies). Using a silhouette image has desirable properties: first, it is a 2D representation and so is limited by the 2D image size (e.g. as opposed to the size of a 3D voxel grid). Second, pixel intensities do not have to be modelled.

3.2 Depth

However, using the silhouette and thereby enforcing the visual hull has the limitation that the network is not forced to predict concavities on the object, as they never appear on the visual hull. The proposed solution to this is to use a multi-task approach. Instead of having the learned representation describe only the silhouette in the new view, the representation must learn additionally to predict the depth of the object in the given views. This enforces that the representation must have a richer understanding of the object, as it must model the concavities on the object as opposed to just the visual hull (which using a silhouette loss imposes). Using a depth image is also a 2D representation, so as with using an image for the silhouette, it is limited by the 2D image size.

4 Implementation

In order to actually implement the proposed approach, the problem is formulated as described in sections 4.1 and 4.2 and a fully convolutional CNN architecture is used, as described in section 4.3.

4.1 Loss function

The loss function is implemented as follows. Given a set of images with their corresponding viewpoints (I_1, θ_1), ..., (I_N, θ_N) a representation x is learned such that x can be used to not only predict the depth in the given views $d_1, ..., d_N$ but also predict the silhouette S at a

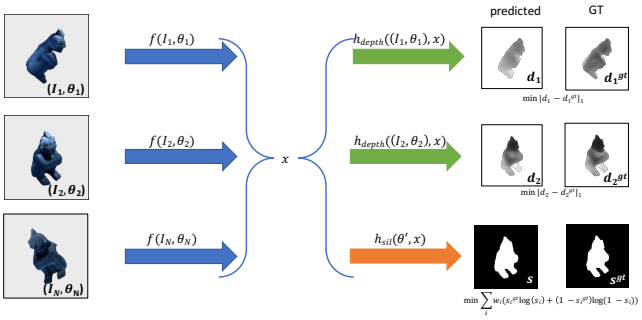


Fig. 2 A diagrammatic explanation of the multi-task loss function used. Given the input images, the images are combined to give a feature vector x which is used by both decoders (denoted in green – depth – and orange – silhouette) to generate the depth predictions for the given views and the silhouette prediction in a new view.

new viewpoint θ'. Moreover, the number of input views (e.g. N) should be changeable at test time such that as N increases then the predictions $d_1, ...d_N, S$ improve.

To do this, the images and their corresponding viewpoints are first encoded using a convolutional encoder f to give a latent representation fv_i. The same encoder is used for all viewpoints giving $f(I_1,\theta_1),...,f(I_N,\theta_N) = fv_1,...,fv_N$. These are then combined to give the latent view-dependent representation x. x is then decoded using a convolutional decoder h_{sil} conditioned on the new viewpoint θ' to predict the silhouette S in the new view. Optionally, x is also decoded via another convolutional decoder h_{depth}, which is conditioned on the given image and viewpoints to predict the depth at the given viewpoints – $d_i = h_{depth}(x, I_i, \theta_i)$. Finally, the binary cross entropy loss is used to compare S to the ground truth S^{gt} and the L_1 loss to compare d_i to the ground truth $d_{i_{gt}}$.

4.2 Improved loss functions

Implementing the loss functions naively as described in section 4.1 is problematic. First, the depth being predicted is the absolute depth, which means the model must guess the absolute position of the object in the scene. This is inherently ambiguous. Second, the silhouette prediction decoder struggles to model the finer detail on the silhouette, instead focusing on the middle of the object which is usually filled.

As a result, both losses are modified. For the depth prediction, the mean of both the ground truth and predicted depth are moved to 0.

The silhouette loss is weighted at a given pixel $w_{i,j}$ based on the Euclidean distance at that point to the silhouette (denoted as $\text{dist}_{i,j}$):

$$w_{i,j} = \begin{cases} \text{dist}_{i,j}, & \text{if } \text{dist}_{i,j} \leq T \\ c & \text{otherwise.} \end{cases} \quad (1)$$

In practice $T = 20, c = 5$. The rationale for the fall-off when $\text{dist}_{i,j} > T$ is due to the fact that most of the objects are centred and have few holes, so modelling the pixels far from the silhouette is easy. Using the fall-off incentivises SiDeNet to correctly model the pixels near the silhouette. Weighting based on the distance to the silhouette models the fact that it is ambiguous whether pixels on the silhouette are part of the background or foreground.

In summary, the complete loss functions are

$$\mathcal{L}_{sil} = \sum_{i,j} w_{i,j}(S^{gt}_{i,j}\log(S_{i,j}) + (1 - S^{gt}_{i,j})\log(1 - S_{i,j})); \quad (2)$$

$$\mathcal{L}_{depth} = \sum_{i=1}^{N} |d_i - d_{i_{gt}}|_1. \quad (3)$$

The loss function is visualised in figure 2. Note that in this example the network's prediction exhibits a concavity in the groove of the sculpture's folded arms.

4.3 Architecture

This section describes the various components of SiDeNet, which are visualised in figure 3 and described in detail in table 10. This architecture takes as input a set of images of size 256×256 and corresponding viewpoints (encoded as $[\sin\theta_i, \cos\theta_i]$ so that $0°, 360°$ map to the same value) and generates depth and silhouette images at a resolution of size 256×256. SiDeNet takes the input image viewpoints as additional inputs because there is no implicit coordinate frame that is true for all objects. For example, a bust may be oriented along the z-axis for one object and the x-axis for another and there is no natural mapping from a bust to a sword. Explicitly modelling the coordinate frame using the input/output viewpoints removes these ambiguities.

SiDeNet is modified to produce a latent 3D representation in SideNet3D, which is visualised in figure 4 and described in section 4.4. This architecture is useful for two reasons. First, it demonstrates that the method of combining multiple views is useful in this scenario as well. Second, it is used to evaluate whether the image representation does indeed allow for more accurate predictions, as the 3D representation necessitates using fewer convolutional transposes and so generates a smaller 57×57 silhouette image.

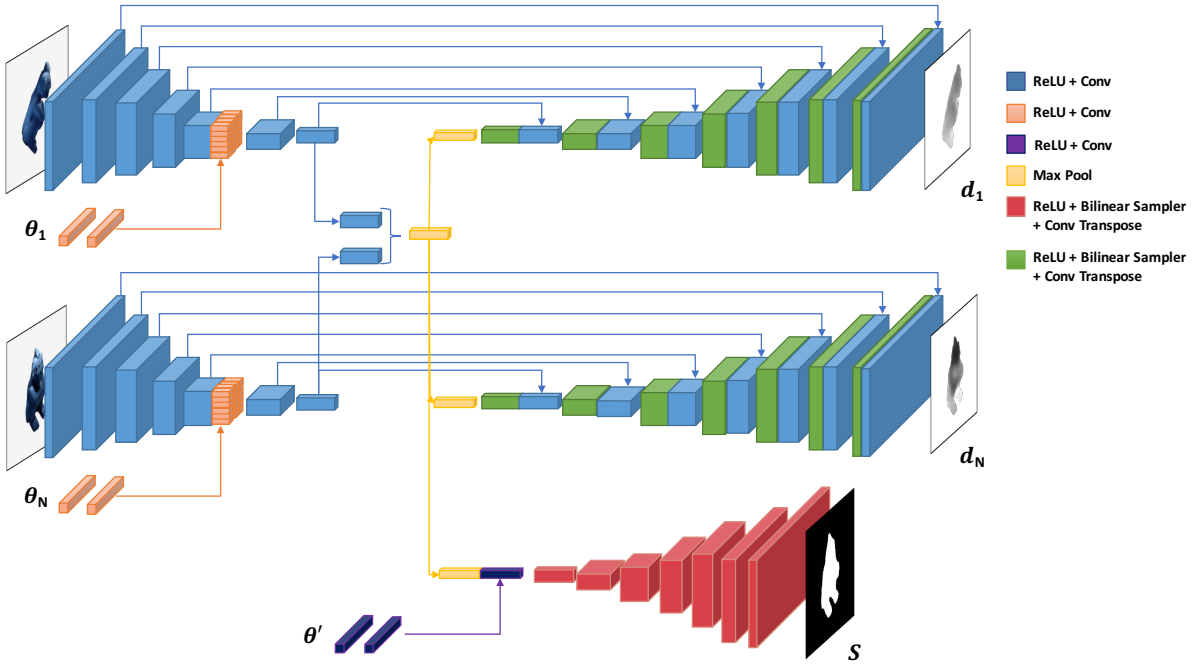


Fig. 3 A diagrammatic overview of the architecture used in SiDeNet. Weights are shared across encoders and decoders (e.g. portions of the architecture having the same colour indicate shared weights). The blue, orange, and purple arrows denote concatenation. The input angles $\theta_1 \dots \theta_N$ are broadcast over the feature channels as illustrated by the orange arrows. The feature vectors are combined to form x (indicated by the yellow block and arrows). This value is then used to predict the depth at the given views $\theta_1 \dots \theta_N$ and the silhouette at a new view θ'. The size of x is invariant to the number of input views N, so an extra view θ_i can be added at test time without any increase in the number of parameters. Please see table 10 for the precise details.

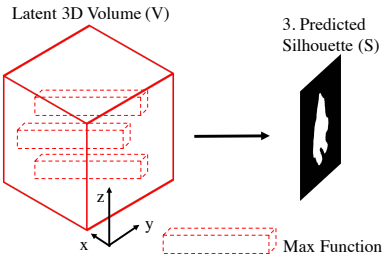


Fig. 4 A diagrammatic overview of the projection in SiDeNet3D. A set of 3D convolutional transposes up-sample from the combined feature vector x to generate the $57\times57\times57$ voxel (V). This is then projected using a max-operation over each pixel location to generate the silhouette in a new view. Please see table 10 for a thorough description of the three different architectures.

Encoder The encoder f takes the given image I_i and theta θ_i and encodes it to a latent representation fv_i. In the case of all architectures, this is implemented using a convolutional encoder, which is illustrated in figure 3. The layer parameters and design are based on the encoder portion of the pix2pix architecture by Isola et al (2017) which is based on the UNet architecture of Ronneberger et al (2015).

Combination function To combine the feature vectors of each encoder, any function that satisfies the following property could be considered: given a set of feature vectors fv_i, the combination function should combine them into a single latent vector x such that for any number of feature vectors, x always has the same number of elements. In particular, an element wise max pool over the feature vectors and an element-wise average pool are considered. This vector x must encode properties of 3D shape useful for both depth prediction and silhouette prediction in a new view.

Decoder (depth) The depth branch predicts the depth of a given image using skip connections (taken from the corresponding input branch) to propagate the higher details. The exact filter sizes are modelled on the pix2pix and UNet networks.

Decoder (silhouette) The silhouette branch predicts the silhouette of a given image at a new viewpoint θ'. The layers are the same as the decoder (depth) branch without the skip connections (as there is no corresponding input view).

Table 1 Overview of the datasets. Gives the number of sculptures in the train/val/test set as well as the number of views per object.

Dataset	Train	Val	Test	# of Views
SketchFab	372	20	33	5
SynthSculptures	77	–	–	5
ShapeNet	4744	678	1356	24

4.4 3D decoder

For SiDeNet3D, the silhouette decoder is modified to generate a latent 3D representation encoded using a voxel occupancy grid. Using a projection layer this grid is projected to 2D, which allows the silhouette loss to be used to train the network in an end-end manner. This is done as follows. First, the decoder is encoded as a sequence of 3D convolutional transposes which generate a voxel of size $V = 57 \times 57 \times 57$ (please refer to appendix A.1 for the precise details). This box is then transformed to the desired output θ' to give V' using a nearest neighbour sampler as described by Jaderberg et al (2015). The box is projected to generate the silhouette in a new view using the max function. As the max function is differentiable, the silhouette loss can be back propagated through this layer and the entire network trained end-to-end.

The idea of using a differentiable projection layer was also considered by Yan et al (2016); Tulsiani et al (2017); Gadelha et al (2016); Rezende et al (2016). (However, we can incorporate additional views *at test time*.)

5 Dataset

Three datasets are used in this work: a large sculpture dataset of scanned objects which is downloaded from SketchFab (2017), a set of scanned sculptures, and a subset of the synthetic ShapeNet objects (Chang et al (2015)). An overview of the datasets are given in table 1. Note that unlike our dataset, ShapeNet consists of object categories for which one can impose a canonical view (e.g. that 0° corresponds to a chair facing the viewer). This allows for methods trained on this dataset to make use of rotations or transformations relative to the canonical view. However, for the sculpture dataset, this property does not exist, necessitating the need of a view-dependent representation for SiDeNet.

Performing data augmentation in 3D is also investigated and shown to increase performance in section 6.2.

5.1 Sculpture datasets

SketchFab: sculptures from SketchFab A set of realistic sculptures are downloaded from SketchFab (the same sculptures as used in Wiles and Zisserman (2017) but different renderings). These are accurate reconstructions of the original sculptures generated by users using photogrammetry and come with realistic textures. Some examples are given in figure 5(a).

SynthSculptures This dataset includes an additional set of 77 sculptures downloaded from TurboSquid[2] using the query *sculpture*. These objects have a variety of realism and come from a range of object classes. For example the sculptures range from low quality meshes that are clearly polygonized to high quality, highly realistic meshes. The object classes range from abstract sculptures to jewellery to animals. Some examples are given in figure 5(b).

Rendering The sculptures and their associated material (if it exists) are rendered in Blender (Blender Online Community (2017)). The sculptures are first resized to be within a uniform range (this is necessary for the depth prediction component of the model). Then, for each sculpture, five images of the sculpture are rendered from uniformly randomly chosen viewpoints between 0° and 120° as the object is rotated about the vertical axis. Three light sources are added to the scene and translated randomly with each render. Some sample sculptures (and renders) for *SketchFab* and *SynthSculptures* are given in figure 5.

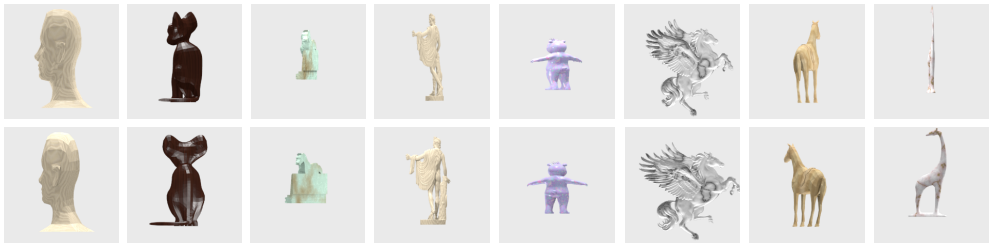
3D augmentation 3D data augmentation is used to augment the two sculpture datasets by modifying the dimensions and material of a given 3D model. The x,y,z dimensions of a model are each randomly scaled from between $[0.5, 1.4]$ of the original dimension. Then a material is randomly chosen from a set of standard blender materials[3]. These materials include varieties of wood, stone, and marble. Finally, the resulting model is rendered from five viewpoints exactly as described above. The whole process is repeated 20 times for each model. Some example renderings using data augmentation for a selection of models from *SynthScultpures* are illustrated in figure 6.

Dataset split The sculptures from *SketchFab* are divided at the sculpture level into train, val, test so that there are 372/20/33 sculptures respectively. All sculptures from *SynthSculptures* are used for training. For

[2] https://www.turbosquid.com/3d-model/
[3] https://www.blendswap.com/blends/view/4867



(a) *SketchFab* dataset. Two sample renderings of seven objects. The first three fall into the train set, the rest into the test set.



(b) *SynthSculpture* dataset. Sample renderings of eight objects. These samples demonstrate the variety of objects, e.g. toys, animals, etc.



(c) *ShapeNet*. Seven sample renderings of the chair subset.

Fig. 5 Sample renderings of the three different datasets. Zoom in for more details. Best viewed in colour.

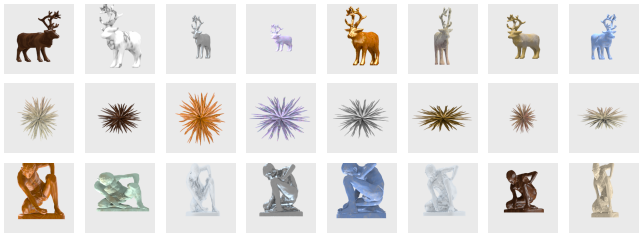


Fig. 6 Seven sample augmentations of three models in the *SynthSculpture* dataset using the 3D augmentation setup described in section 5.1. These samples demonstrate the variety of materials, sizes and viewpoints for a given 3D model using the 3D data augmentation method.

a given iteration during train/val/test, a sculpture is randomly chosen from which a subset of the 5 rendered views is selected.

5.2 ShapeNet

ShapeNet (Chang et al (2015)) is a dataset of synthetic objects divided into a set of semantic classes. To compare this work to that of Yan et al (2016), their subdivision, train/val/test split and renderings of the ShapeNet chair subset are used. Their rendered synthetic objects are rendered under simple lighting conditions at fixed 15° intervals about the vertical axis for each object to give a total of 24 views per object. We additionally collect depth maps for each render using the extrinsic/intrinsic parameters of Yan et al (2016). Some example renderings are given in figure 5(c). Again at train/val/test time, a sculpture is randomly chosen and a subset of this sculpture's 24 renders is chosen.

6 Experiments

This section first evaluates the design choices: the utility of using the data augmentation scheme is demonstrated in section 6.2, the effect of the different architectures in section 6.3, the multi-task loss in section 6.4, and the effect of the choice of θ' in section 6.8. Second it evaluates the method of combining multiple views: sections 6.5 and 6.6 demonstrate how increasing the number of views at test time improves performance on the *Sculpture* dataset irrespective of whether the input/output views are photometrically consistent. Section 6.7 demonstrates that the approach works on ShapeNet and section 6.9 evaluates the approach in 3D. SiDeNet's ability to perform new view synthesis is exhibited in section 7 as well as its generalisation capability to real images. Finally, the method by which

SiDeNet can encode a joint embedding of shape and viewpoint is investigated in section 8.

6.1 Training setup

The networks are written in pytorch (Paszke et al (2017)) and trained with SGD with a learning rate of 0.001, momentum of 0.9 and a batch size of 16. They are trained until the loss on the validation set stops improving or for a maximum of 200 iterations, whichever happens first. The tradeoff between the two losses – $\mathcal{L} = \lambda_{depth}\mathcal{L}_{depth} + \lambda_{sil}\mathcal{L}_{sil}$ – is set such that $\lambda_{depth} = 1$ and $\lambda_{sil} = 1$.

6.1.1 Evaluation measure

The evaluation measure used is the intersection over union (IoU) error for the silhouette, L_1 error for the depth error, and chamfer distance for the error when evaluating in 3D. The IoU for a given predicted silhouette S and ground truth silhouette $\bar{S}$ is evaluated as $\frac{\sum_{x,y}(I(S)\cap I(\bar{S}))}{\sum_{x,y}(I(S)\cup I(\bar{S}))}$ where I is an indicator function and equals 1 if the pixel is a foreground pixel, else 0. This is then averaged over all images to give the mean IoU.

The L_1 loss is simply the average over all foreground pixels: $L_1 = \frac{1}{N}\sum_{p_x} |d^{pred}_{p_x} - d^{gt}_{p_x}|_1$ where p_x is a foreground pixel and N the number of foreground pixels. Note that the predicted and ground truth depth are first normalised by subtracting off the mean depth. This is then averaged over the batch. When there are multiple input views, the depth error is only computed for the first view, so the comparison across increasing numbers of views is valid.

The chamfer distance used is the symmetrized version. Given the ground truth point cloud g and the predicted one p, then the error is $CD = \frac{1}{N}\sum_{i=1}^{N} \min_j |g_i - p_j|^2 + \frac{1}{M}\sum_{i=1}^{M} \min_j |g_j - p_i|^2$.

6.1.2 Evaluation Setup

Unless otherwise stated, the results are for the max-pooling version of SiDeNet, with input/output view size 256×256, trained with 2 distinct views, data augmentation of both datasets (section 6.2), $\lambda_{depth} = 1$ and $\lambda_{sil} = 1$, and the improved losses described in section 4.2.

6.2 The effect of the data augmentation

First, the effect of the 3D data augmentation scheme is considered. The results for four methods trained with varying amounts of data augmentation (described in section 5.1) are reported in table 2 and demonstrate the benefit of using the 3D data augmentation scheme. (These are trained with the non-improved losses.) Using only 2D modifications was tried but not found to improve performance.

Table 2 Effect of data augmentation. This table demonstrates the utility of using 3D data augmentation to effectively enlarge the number of sculptures being trained with. *SketchFab* is always used and sometimes augmented (denoted by *Augment*). *SynthSculpture* is sometimes used (denoted by *Used*) and sometimes augmented. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU.

SketchFab	*SynthSculpture*		L_1 Depth Error	Silhouette IoU
Augment?	Used?	Augment?		
✗	✗	–	0.210	0.643
✓	✗	–	0.202	0.719
✗	✓	✗	0.209	0.678
✓	✓	✓	0.201	0.724

6.3 Ablation study of the different architectures

This section compares the performance of $\text{SiDeNet}_{57\times57}$, SiDeNet3D, and SiDeNet on the silhouette/depth prediction tasks, as well as using average vs max-pooling. SiDeNet/SiDeNet3D are described in section 4.3. $\text{SiDeNet}_{57\times57}$ modifies SiDeNet to generate a 57×57 silhouette (for the details for all architectures please refer to appendix A.1). It additionally compares the simple version of the loss functions, described in section 4.1 to the improved version described in section 4.2. Finally the performance of predicting the mean depth value is given as a baseline. See table 3 for the results.

These results demonstrate that while the difference in the pooling function in terms of results is minimal, our improved loss functions improve performance. Weighting more strongly the more difficult parts of the silhouette (e.g. around the boundary) can encourage the model to learn a better representation.

Finally, $\text{SiDeNet}_{57\times57}$ does worse than SiDeNet for both the L_1 loss and the silhouette IoU loss. While in this case the difference is small, as more data is introduced and the predictions become more and more accurate, the benefit of using a larger image/representation is clear. This is demonstrated by the chairs on ShapeNet in section 6.7.

Table 3 Ablation study of the different architectures, which vary in size and complexity. $_{basic}$ refers to using the standard L_1 and binary cross entropy loss without the improvements described in section 4.2. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU. The sizes denote the size of the corresponding images (e.g. 256×256 corresponds to an output image of this resolution).

Model	Input Size	Output Size	Pooling?	Improved loss?	Depth $L_{1_{256\times256}}$ error	Silhouette $\text{IoU}_{256\times256}$
SiDeNet$_{\text{basic}}$	256×256	256×256	max	✗	0.201	0.724
SiDeNet	256×256	256×256	max	✓	0.181	0.739
SiDeNet	256×256	256×256	avg	✓	0.189	0.734
SiDeNet$_{57 \times 57_{\text{basic}}}$	256×256	57×57	max	✗	–	0.723
SiDeNet$_{57 \times 57}$	256×256	57×57	max	✓	0.195	0.734
SiDeNet3D	256×256	57×57	max	✓	0.182	0.733
Baseline: $z = c$	–	–	–	–	0.223	–

Table 4 Effect of the multi-task loss. This table demonstrates the effect of the multi-task loss. As can be seen, using both losses does not negatively affect the performance of either task. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU.

Loss Function	λ_{depth}	λ_{sil}	Depth L_1 Error	Silhouette IoU
Silhouette & Depth	1	1	0.181	0.739
Silhouette	–	–	–	0.734
Depth	–	–	0.178	–

Table 5 Effect of incorporating additional views at test time. This architecture was trained with one, two, or three views. These results demonstrate how additional views can be dynamically incorporated at test time and results on both depth and silhouette measures improve. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU.

Pooling?	# Views (Train)	# Views (Test)	L_1 Depth Error	Silhouette IoU
max	1	1	0.206	0.702
max	1	2	0.210	0.712
max	1	3	0.209	0.716
max	2	1	0.204	0.694
max	2	2	0.181	0.739
max	2	3	0.170	0.751
avg	2	1	0.197	0.715
avg	2	2	0.192	0.725
avg	2	3	0.189	0.732
max	3	1	0.198	0.706
max	3	2	0.172	0.753
max	3	3	0.162	0.766

6.4 The effect of using $\mathcal{L}_{depth}$ and $\mathcal{L}_{sil}$

Second, the effect of the individual components of the multi-task loss is considered. The multi-task loss enforces that the network learns a richer 3D representation; the network must predict concavities in order to perform well at predicting depth and it must learn about the visual hull of the object in order to predict silhouettes at new viewpoints. As demonstrated in table 4, using the multi-task loss does not negatively affect the prediction accuracy as compared to predicting each component separately. This demonstrates that the model is able to represent both aspects of shape at the same time.

Some visual results are given in figures 11 and 12. Example (b) in figure 12 demonstrates how the model has learned to predict concavities, as it is able to predict grooves in the relief.

6.5 The effect of increasing the number of views

Next, the effect of increasing the number of input views is investigated with interesting results.

For SiDeNet, as with SilNet, increasing the number of views improves results over all error metrics in table 5. Some qualitative results are given in figure 7. It is interesting to note that not only does the silhouette performance improve given additional input views but so does the depth evaluation metric. So incorporating additional views improves the depth prediction for a given view using only the latent vector x.

A second interesting point is that training with more views can predict better than training with fewer numbers of views – e.g. training with three views and testing on one or two views does better than training on two views and testing on two or training on one view and testing on one view. It seems that when training with additional views and testing with a smaller number, the network can make use of information learned from the additional views. This demonstrates the generalisability of the SiDeNet architecture.

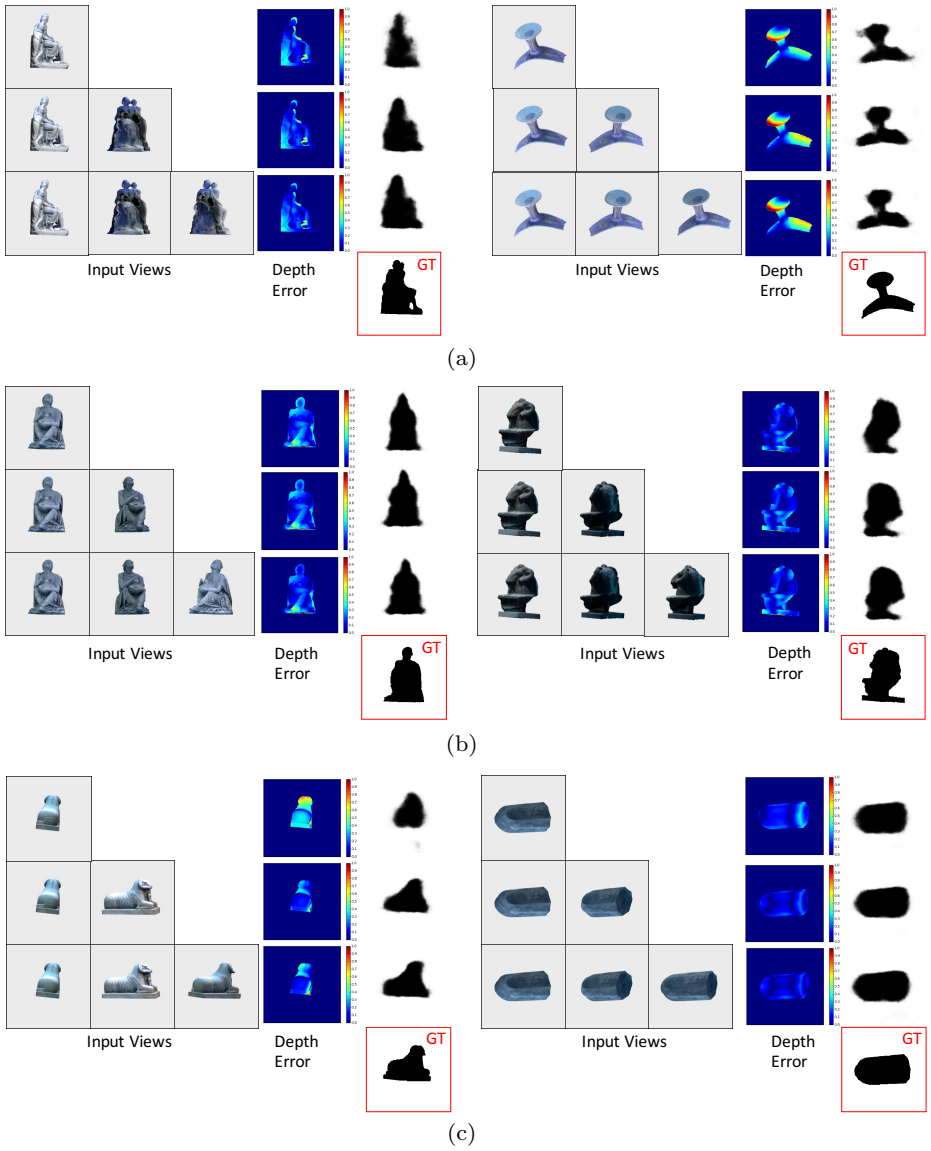


Fig. 7 Qualitative results for increasing the number of input views on SiDeNet. SiDeNet's depth and silhouette predictions are visualised as the number of input views is increased. To the left are the input views, the centre gives the depth prediction for the first input view, and the right gives the predicted silhouette for each set of input views. The silhouette in the red box gives the ground truth silhouette. The scale on the side gives the error in depth – blue means the depth prediction is perfectly accurate and red that the prediction is off by 1 unit. (The depth error is clamped between 0 and 1 for visualisation purposes.) As can be seen, performance improves with additional views. This is most clearly seen for the ram in (c).

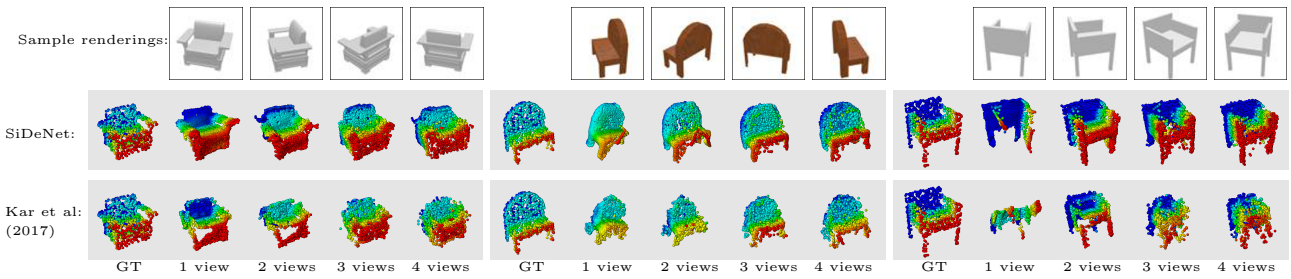


Fig. 8 Comparison of multi-view methods on ShapeNet. Renderings of the given chair are given in the top row, followed by SiDeNet's and Kar et al (2017)'s predictions. For each chair, for each row, the point clouds from left to right show the ground truth followed by the predictions for one, two, three, and four views respectively. The colour denotes the z value. As can be seen SiDeNet's predictions are higher quality than those of Kar et al (2017) for these examples.

Table 6 The effect of using non-photometrically consistent inputs. These results demonstrate that SiDeNet trained with views of an object with the same texture generalises *at runtime* to incorporating views of an object with differing textures. Additional views can be dynamically incorporated at test time and results on both depth and silhouette measures improve. The model is trained with 2 views. The models are evaluated on the test set of *SketchFab*. Lower is better for L_1 and higher is better for IoU.

Views have the same texture?	# Views (Test)	L_1 Depth Error	Silhouette IoU
✓	1	0.165	0.739
✓	2	0.142	0.778
✓	3	0.139	0.785
✗	1	0.164	0.738
✗	2	0.143	0.777
✗	3	0.139	0.785

6.6 The effect of non-photometrically consistent inputs

A major benefit of SiDeNet is it does not require photometrically consistent views: provided the object is of the same shape, then the views may vary in lighting or material. While the sculpture renderings used already vary in lighting conditions across different views (section 5), this section considers the extreme case: how does SiDeNet perform when the texture is modified in the input views. To perform this comparison, SiDeNet is tested on the sculpture dataset with a randomly chosen texture for each view (see figure 6 for some sample textures demonstrating the variety of the 20 textures). It is then tested again on the same test set but with the texture fixed across all input views. The results are reported in table 6.

Surprisingly, with no additional training, SiDeNet performs nearly as well when the input/output views have randomly chosen textures. Moreover, performance improves given additional views. The network appears to have learned to combine input views with varying textures without being explicitly trained for this. This demonstrates a real benefit of SiDeNet over traditional approaches – the ability to combine multiple views of an object for shape prediction *without* requiring photometric consistency.

6.7 Comparison on ShapeNet

SiDeNet is compared to Perspective Transformer Nets by Yan et al (2016) by training and testing on the chair subset of the ShapeNet dataset. The comparison demonstrates three benefits of our approach: the ability to incorporate multiple views, the benefit of our 3D data augmentation scheme, and the benefits of staying in 2D. This is done by comparing the accuracy of SiDeNet's predicted silhouettes to those of Yan et al (2016). Their model is trained with the intention of using it for 3D shape prediction, but we focus on the 2D case here to demonstrate that using an image representation means that, with the same data, we can achieve better prediction performance in the image domain, as we are not limited by the latent voxel resolution. To compare the generated silhouettes, their implementation of the IoU metric is used: $\frac{\sum_{x,y} I(S_{x,y}) \times \bar{S}_{x,y}}{\sum_{x,y} (I(S_{x,y}) + \bar{S}_{x,y}) > 0.9}$.

Multiple setups for SiDeNet are considered: fine-tuning from the model trained on the sculptures with data augmentation (e.g. both in table 1), with/without the improved loss function and for multiple output sizes. To demonstrate the benefits of the SiDeNet architecture, SiDeNet is trained only with the silhouette loss, so both models are trained with the exact same information. The model from Yan et al (2016) is fine-tuned from a model trained for multiple ShapeNet categories. The results are reported in table 7.

These results demonstrate the benefits of various components of SiDeNet, which outperforms Yan et al (2016). *First*, using a 2D resolution means a much larger image segmentation can be used to train the network. As a result, much better performance can be obtained (e.g. $\text{SiDeNet}_{256 \times 256_{\text{basic}}}$ has much better performance than $\text{SiDeNet}_{57 \times 57_{\text{basic}}}$). *Second*, the improved, weighted loss function for the silhouette (section 4.2) improves performance further. *Third*, fine-tuning a model trained with the 3D sculpture augmentation scheme gives an additional small boost in performance. *Finally*, using additional views improves results for all versions of SiDeNet. Some qualitative results are given in figure 10.

6.8 The effect of varying θ'

In order to see how well SiDeNet can extrapolate to new angles (and there by how much it has learned about the visual hull), the following experiment is performed on ShapeNet. SiDeNet is first trained with various ranges of θ', θ_i. For example if the range is $[15^\circ \dots 120^\circ]$, then all randomly selected input angles θ_i and θ' are constrained to be within this range during training. At test time, a random chair is chosen and the silhouette IoU error evaluated for each target viewpoint θ' in the full range (e.g. $[15^\circ \dots 360^\circ]$), but the input angles θ_i are still constrained to be in the constrained range (e.g. $[15^\circ \dots 120^\circ]$). This evaluates how well the model extrapolates to unseen viewpoints at test time and how well it has learned about shape. If the model was perfect, then there would be no performance degradation

Table 7 Comparison to Perspective Transformer Nets (PTNs) (Yan et al (2016)) on the silhouette prediction task on the chair subset of ShapeNet. Their model is first trained on multiple ShapeNet categories and fine-tuned on the chair subset. SiDeNet is optionally first trained on the Sculpture dataset or trained directly on the chair subset. As can be seen, SiDeNet outperforms PTN given one view and improves further given additional views. These results also demonstrate the utility of various components of SiDeNet: using a larger 256×256 image to train the silhouette prediction task and using the improved, weighted loss function. It is also interesting to note that pre-training with the complex sculpture class gives a small boost in performance (e.g. it generalises to this very different domain of chairs). The value reported is the mean IoU metric for the silhouette; higher is better.

		Number of views tested with				
	Pre-training	1	2	3	4	5
Yan et al (2016)	ShapeNet	0.797	–	–	–	–
SiDeNet	Sculptures	**0.831**	**0.845**	**0.850**	**0.852**	**0.853**
SiDeNet	–	0.826	0.843	0.848	0.850	0.851
$\text{SiDeNet}_{256 \times 256_{\text{basic}}}$	–	0.814	0.831	0.835	0.837	0.837
$\text{SiDeNet}_{57 \times 57_{\text{basic}}}$	–	0.775	0.791	0.795	0.796	0.795

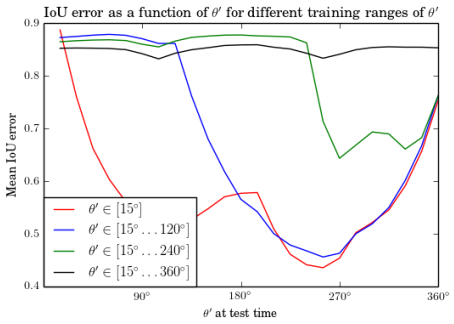


Fig. 9 The effect of varying the range of θ' used at train time on the IoU error at test time.

as θ' moved out of the constrained range used to train the model. The results are given in figure 9. As can be seen (and would be expected), for various training ranges the performance degrades as a function of how much θ' differs from the range used to train the model. The model is able to extrapolate outside of the training range, but the more the model must extrapolate, the worse the prediction.

6.9 Comparison in 3D

We additionally evaluate SiDeNet's 3D predictions and consider the two cases: using the depth maps predicted by SiDeNet and the voxels from SiDeNet3D.

SiDeNet The depth maps are compared to those predicted using the depth map version of Kar et al (2017) in table 8. This comparison is only done on ShapeNet as for the *Sculpture* dataset we found it was necessary to subtract off the mean depth to predict high quality depth maps (section 4.2). However, for ShapeNet there is less variation between the chairs so this is not necessary. As a result SiDeNet is trained with 2 views, the improved silhouette loss but the depth predicted is the absolute depth. The comparison is performed as follows for both methods. For each chair in the test set an initial view is chosen and the depth back-projected using the known extrinsic/intrinsic camera parameters. Then for each additional view, the initial views are chosen by sampling evenly around the z-axis (e.g. if the first view is at 15°, then two views would be at $15°, 195°$ and three views at $15°, 195°, 255°$) and the depth again back-projected to give a point cloud. 2500 points are randomly chosen from the predicted point cloud and aligned using ICP (Besl and McKay (1992)) with the ground truth point cloud. This experiment evaluates the method of pooling information in the two methods and demonstrates that SiDeNet's global method of combining information performs better than that of Kar et al (2017) which combines information along projection rays. Some qualitative results are given in figure 8.

SiDeNet3D SiDeNet3D is trained with 2 views and the improved losses. The predicted voxels from the 3D projection layer are extracted and marching cubes used to fit a mesh over the iso-surface. The threshold value is chosen on the validation set. A point cloud is extracted by randomly sampling from the resulting mesh.

SiDeNet3D is compared to a number of other methods in table 9 for the *Sculpture* dataset. For SiDeNet3D and all baselines models, 2500 points are randomly chosen from the predicted point cloud and aligned with the ground truth point cloud using ICP. The resulting point cloud is compared to the ground truth point cloud by reporting the chamfer distance (CD). As can be seen, the performance of our method improves as the number of input views increases.

Additionally, SiDeNet3D performs better than other baseline methods on the *Sculpture* dataset in table 9 which demonstrates the utility of explicitly encoding the input viewpoint and thereby representing

the coordinate frame of the object. We note again that there is no canonical coordinate frame and the input viewpoint does not align with the output shape, so just predicting the 3D without allowing the network to learn the transformation from the input viewpoint to the 3D (as done in all the baseline methods) leads to poor performance.

Baselines The baseline methods which do not produce point clouds are converted as follows. To convert Yan et al (2016) to a point cloud, marching cubes is used to fit a mesh over the predicted voxels. Points are then randomly chosen from the extracted mesh. To convert Tatarchenko et al (2016) to a point cloud, the model is used to predict depth maps at $[0°, 90°, 180,°, 270°]$. The known intrinsic/extrinsic camera parameters are used to back-project the depth maps. The four point clouds are then combined to form a single point cloud.

7 Generating new views

Finally SiDeNet's representation can be qualitatively evaluated by performing two tasks that require new view generation: rotation and new view synthesis.

7.1 Rotation

As SiDeNet is trained with a subset of views for each dataset (e.g. only 5 views of an object from a random set of viewpoints in $[0°, 120°]$ for the *Sculpture* dataset and 24 views taken at 15° intervals for ShapeNet), the angle representation can be probed by asking SiDeNet to predict the silhouette as the angle is continuously varied within the given range of viewpoints. Given a fixed input, if the angle is varied continuously, then the output should similarly vary continuously. This is demonstrated in figure 10 for both the *Sculpture* and ShapeNet databases.

7.2 New view synthesis

Using the predicted depth, new viewpoints can be synthesised, as demonstrated in figure 11. This is done by rendering the depth map of the object using Open3D (Zhou et al (2018)) as a point cloud at the given viewpoint and at a 45° rotation. At both viewpoints the object is rendered in three ways: using a textured point cloud, relighting the textured point cloud, and rendering the point cloud using the predicted z value.

7.3 Real Images

Finally, the generalisability of what SiDeNet has learned is tested on another dataset of real images of sculptures, curated by Zollhöfer et al (2015). The images of two sculptures (augustus and relief) are taken. The images are segmented and padded such that the resulting images have the same properties as the *Sculpture* dataset (e.g. distance of sculpture to the boundary and background colour). The image is then input to the network with viewpoint 0°. The resulting prediction is rendered as in section 7.2 at multiple viewpoints and under multiple lighting conditions in figure 12. This figure demonstrates that SiDeNet generalises to real images, even though SiDeNet is trained only on synthetic images and for a comparatively small (only $\approx$ 400) sculptures. Moreover these real images have perspective effects, yet SiDeNet generalises to these images, producing realistic predictions.

8 Explainability

This section delves into SiDeNet, attempting to understand how the network learns to incorporate multiple views. To this end, the network is investigated using two methods. The first considers how well the original input images can be reconstructed given the angles and feature encoding x. The second considers how well the original input viewpoints θ_i can be predicted as a function of the embedding x and what this implies about the encoding. This is done for both the max and average pooling architectures.

8.1 Reconstruction

The first investigation demonstrates that the original input images can be relatively well reconstructed given only the feature encoding x and the input views. These reconstructions in figure 13 demonstrate that x must hold some viewpoint and image information.

To reconstruct the images, the approach of Mahendran and Vedaldi (2015) is followed. Two images and their corresponding viewpoints, are input to the network and a forward pass computed. Then the combined feature vector x is extracted (so it contains the information from the input views and their viewpoints). The two images are reconstructed, starting from noise, by minimizing a cost function consisting of two losses: the first loss, the $\mathcal{L}_{MSE}$ error, simply says that the two reconstructed images when input to the network, should give a feature vector x' that is the same as x. The second

Table 8 CD (× 100) on the *ShapeNet* dataset. The models evaluated on depth predict a depth map which is back-projected to generate a 3D point cloud.

Model	Trained with:	Evaluation is on:	Number of views tested with 1	2	3	4	6
SiDeNet	Silhouettes + Depth	Depth	**1.47**	**0.72**	**0.62**	**0.59**	**0.58**
Kar et al (2017)	Depth	Depth	1.73	0.82	0.71	0.67	0.65

Table 9 CD (× 100) on the *Sculptures* dataset. The models evaluated on depth predict a depth map which is back-projected to generate a 3D point cloud. The models evaluated on 3D are compared using the explicitly or implicitly learned 3D.

Model	Trained with	Evaluation is on:	Number of views tested with 1	2	3
SiDeNet3D	Silhouettes + Depth	3D	**0.87**	**0.82**	**0.81**
Kar et al (2017)	Depth	Depth	2.15	1.38	1.15
Tatarchenko et al (2016)	Depth	Depth	1.97	–	–
Yan et al (2016)	Silhouettes	3D	1.26	–	–
Groueix et al (2018)	3D	3D	1.23	–	–

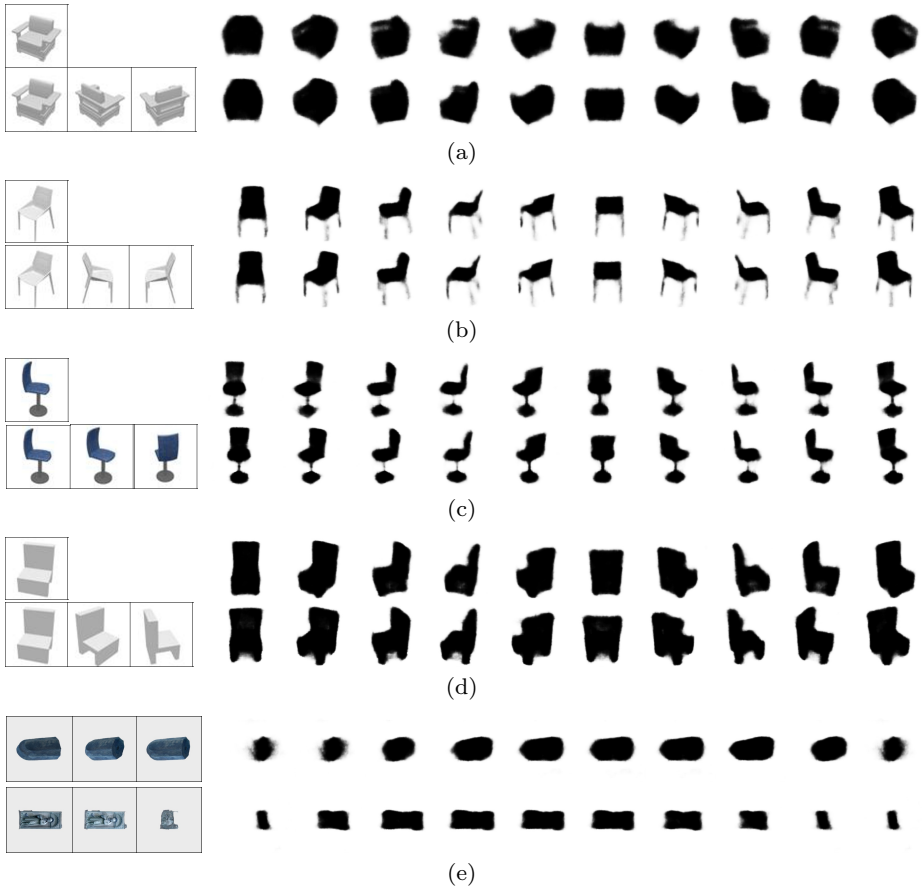


Fig. 10 Qualitative results for rotating an object using the angle embedding of θ'. As the angle θ' is rotated from $[0°, 360°]$ while the input images and viewpoints are kept fixed, it can be seen that the objects rotate continuously for ShapeNet ((a)-(d)) and the *Sculpture* database (e). Additionally, the results for ShapeNet improve given additional input views. For example, in (d), the base of the chair is incorrectly predicted as solid given one view but correctly predicted given additional views.

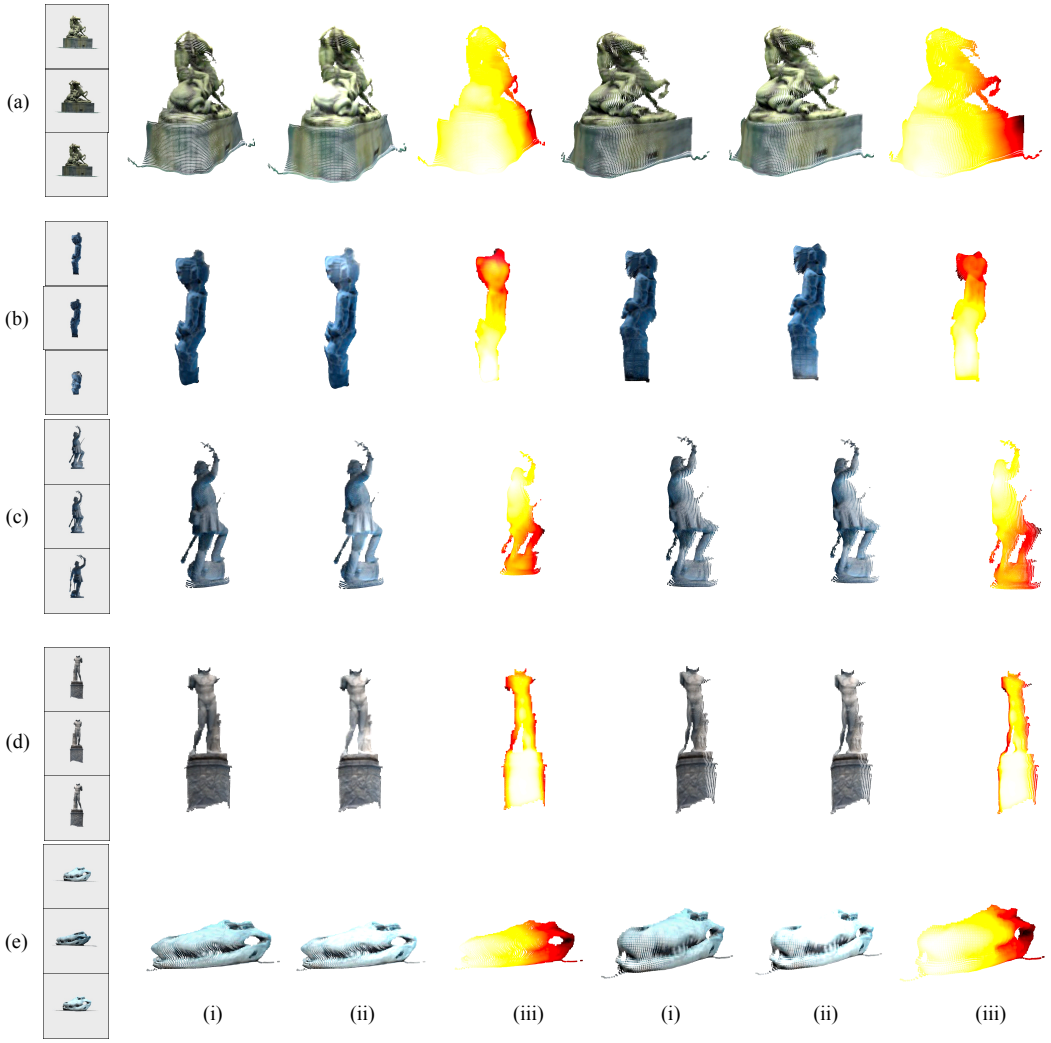


Fig. 11 This figure demonstrates how new views of a sculpture can be synthesised. For each sculpture the input views are shown to the left. The sculpture is then rendered at two viewpoints. At each viewpoint, three renderings are shown: (i) the rendered, textured point cloud, (ii) the point cloud relit and (iii) the depth cloud rendered by using the z-value for the colour (e.g. dark red is further away and yellow/white nearer). Zoom in for details.

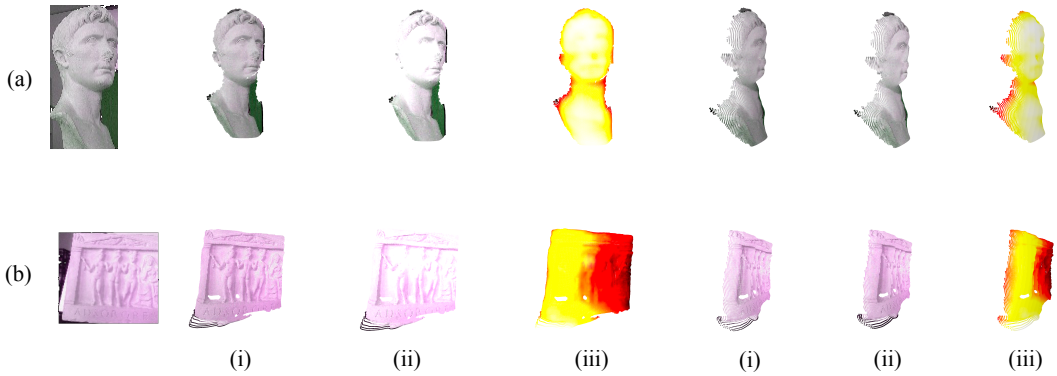


Fig. 12 SiDeNet's predictions for real images. This figure demonstrates how SiDeNet generalises to real images. For each sculpture the input view (before padding and segmentation) is shown to the left. The predicted point cloud is then rendered at two viewpoints. At each viewpoint, three renderings are shown: (i) the rendered, textured point cloud, (ii) the point cloud relit and (iii) the depth cloud rendered by using the z-value for the colour (e.g. dark red is further away and yellow/white nearer). Zoom in for details.

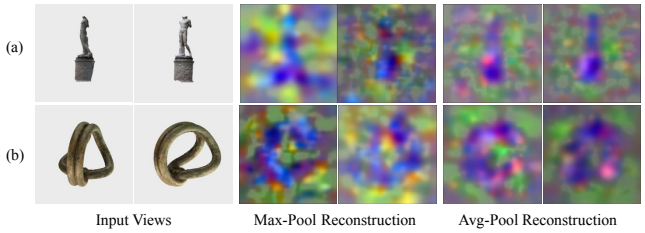


Fig. 13 Reconstruction of the original input images for max/avg pooling architectures. The ability to propagate view and viewpoint information through the network is demonstrated by the fact that the input images can be reconstructed given the latent feature vector and input angles using the approach of Mahendran and Vedaldi (2015).

loss, the total variation regulariser $\mathcal{L}_{TV}$ (as in Mahendran and Vedaldi (2015) and Upchurch et al (2017)), states that the reconstructed images should be smooth.

$$\mathcal{L}_{MSE} = \sum_i (x_i - x_i')^2 \quad (4)$$

$$\mathcal{L}_{TV} = \sum_{i,j} ((I_{i,j+1} - I_{i,j})^2 + (I_{i+1,j} - I_{i,j})^2)^{\beta/2} \quad (5)$$

This gives the total loss $\mathcal{L} = \mathcal{L}_{MSE} + \lambda_{TV} * \mathcal{L}_{TV}$. Here, β, λ_{TV} are chosen such that $\beta = 2$ and $\lambda_{TV} = 0.001$. The cost function is optimized using SGD (with momentum 0.975 and learning rate 1, which is decreased by a factor of 0.1 at each 1000 steps).

8.2 Analysis of feature embeddings

In the reconstructions above, it seems that some viewpoint information is propagated through the network, despite the aggregation function. Here, we want to understand precisely how this is done. In order to do so, the following experiment is conducted: how well can the various viewpoints (e.g. $\theta_1 ... \theta_N$) be predicted for a given architecture from the embedding x. If the hypothesis – that the embedding x encodes viewpoint – is correct, then these viewpoints should be accurately predicted.

As a result, x is considered to determine how much of it is viewpoint-independent and how much of it is viewpoint-dependent. This is done by using each hidden unit in x to predict the viewpoint θ_1 using ordinary least squares regression (Friedman et al (2001)) (only θ_1 is considered as x is invariant to the input ordering). Training pairs are obtained by taking two images with corresponding viewpoints θ_1 and θ_2, passing them through the network and obtaining x.

The p-value for each hidden unit is computed to determine whether there is a significant relation between the hidden unit and the viewpoint. If the p-value is insignificant (i.e. it is large, > 0.05) then this implies that the hidden unit and viewpoint are not related, so it

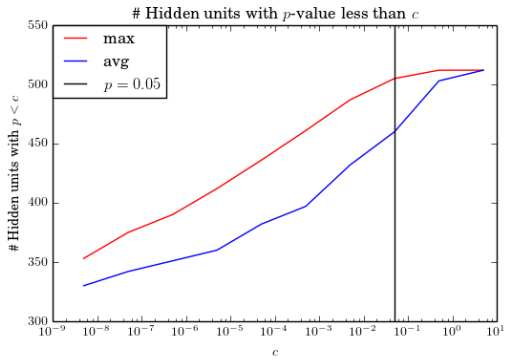


Fig. 14 Visualises the relation between the individual hidden units and the viewpoint. Each hidden unit is used in a separate regression to predict the viewpoint. The p-value for each hidden unit is computed and for a given set of values c, the number of hidden units with a p-value $< c$ is plotted. This demonstrates that the majority of hidden units in both architectures are correlated with the viewpoint. For the max architecture, 98% of the hidden units have $p < 0.05$ and for the avg pool architecture 90%.

contains viewpoint-independent information (presumably shape information). The number of hidden units with p-value less than c, as c is varied, is visualised in figure 14 for both architectures. As can be seen, more than 80% of the hidden units for both architectures are significantly related to the viewpoint.

Since so many of the hidden units have a significant relation to the viewpoint, they would be expected to vary as a function of the input angle. To investigate this, the activations of the hidden units are visualised as a function of the angle θ_1. For two objects, all input values are kept fixed (e.g. the images and other viewpoint values) except for θ_1 which is varied between 0° and 360°. A subset of the hidden units in x are visualised as θ_1 is varied in figure 15. As can be seen, the activation either varies in a seemingly sinusoidal fashion – it is maximised at some value for θ_1 and decays as θ_1 is varied – or it is constant.

Moreover, the activations are not the same if the input images are varied. This implies that the hidden units encode not just viewpoint but also viewpoint-dependent information (e.g. shape – such as the object is tall and thin at 90°). This information is aggregated over all views with either aggregation method. The aggregation method controls whether the most 'confident' view (e.g. if using max) is chosen or all views are considered (e.g. avg). Finally, this analysis demonstrates the utility of encoding the input viewpoints in the architecture. When generating the silhouette and depth at a new or given viewpoint, these properties can be easily morphed into the new view (e.g. if the new view-

point is at 90° then components nearer 90° can be easily considered with more weight by the model).

8.3 Discussion

In this section, to understand what two versions of SiDeNet – avg and max – have learned, two questions have been posed. How well can the original input images be reconstructed from the angles and latent vector x? How is x encoded such that views can be aggregated and that with more views, performance improves? The subsequent analysis has not only demonstrated that the original input views can be reconstructed given the viewpoints and x but has also put forward an explanation for how the views are aggregated: by using the hidden units to encode shape and viewpoint together.

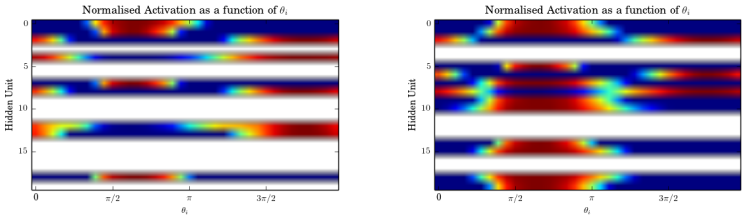
9 Summary

This work has introduced a new architecture SiDeNet for learning about 3D shape, which is tested on a challenging dataset of 3D sculptures with a high variety of shapes and textures. To do this a multi-task loss is used; the network learns to predict the depth for the given views and the silhouette at a new view. This loss has multiple benefits. First, it enforces that the network learns a complex representation of shape, as predicting the silhouette enforces that the network learns about the visual hull of the object and predicting the depth that the network learns about concavities on the object's surface. Second, using an image-based representation is beneficial, as it does not limit the resolution of the generated model; this benefit is demonstrated on the ShapeNet dataset. The trained network can then be used for various applications, such as new view synthesis and can even be used directly on real images.

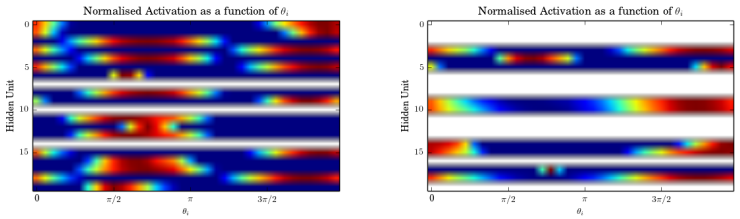
The second benefit of the SiDeNet architecture is the view-dependent representation and the ability to generalise over additional views at test-time. Using a view-dependent representation means that no implicit assumptions need to be made about the nature of the 3D objects (e.g. that there exists a canonical orientation). Additionally, SiDeNet can leverage additional views at test time and results (both silhouette and depth) improve with each additional view, *even* when the views are not photometrically consistent.

While the architecture is able to capture a wide variety of shapes and styles as demonstrated in our results, it is most likely that SiDeNet would improve given more data. However, despite the sculpture dataset being small compared to standard deep learning datasets, it is interesting that SiDeNet can be used to boost performance on a very different synthetic dataset of chairs and predict depth, out-of-the-box, on real sculpture images.

Acknowledgements Thank you to Andrew Fitzgibbon and the anonymous reviewers for their useful comments. This work was funded by an EPSRC studentship and EPSRC Programme Grant Seebibyte EP/M013774/1.



(a) Shows the activation for a subset of hidden units for the avg-pooling architecture for two different sets of input images (left and right).



(b) Shows the activation for a subset of hidden units for the max-pooling architecture for two different sets of input images (left and right).

Fig. 15 Visualisation of the activation of hidden units as a function of θ_i for the two architectures. θ_i is varied between $0°, 360°$ and all other values kept constant. Each hidden unit is normalised to between 0 and 1 over this sequence of θ_i and visualised. This figure demonstrates two things: that the activation is a continuous, smooth function of θ_i or constant (visualised as white in the figure). Second, it demonstrates that the hidden units activated are based on the input views, as they vary from view to view. This implies that the hidden units encode viewpoint dependent information (e.g. object properties and the associated viewpoint).

A Additional Architectural Details

A.1 2D architecture

Table 10 gives additional information about the 2D architectures used. There are two variations. The first takes a 256×256 architecture and generates silhouette and depth images of size 256×256. The second stays in 2D and modifies the silhouette decoder to generate a smaller silhouette of size 57×57.

A.2 3D decoder

The third architecture modifies the silhouette decoder to generate a latent 3D representation which projects to a silhouette of size 57×57 (the encoder is the same as for the 2D architectures). The 3D decoder is composed of the following set of 3D convolutional transposes and ReLU units. ConvT3D(256,3,2) $\rightarrow$ ReLU $\rightarrow$ ConvT3D(128,3,2) $\rightarrow$ ReLU $\rightarrow$ ConvT3D(64,3,2) $\rightarrow$ ReLU $\rightarrow$ ConvT3D(1,4,2). ConvT3D(c,k,s) denotes a 3D convolutional transpose layer with c output channels, a kernel size k and stride s. The resulting $57 \times 57 \times 57$ voxel is finally transformed as described in section 4.4.

Table 10 Overview of the different architectures. The colours correspond to figure 3. The part in orange corresponds to the angle encoding and the part in blue the image encoding. These are then concatenated at layer **6** by broadcasting the angle encoding across the spatial dimensions of the image tensor to which it is supposed to be concatenated. Layer type *Conv* denotes convolution followed by an *Leaky ReLU (0.2)* layer. Layer type *Upsamp* denotes a sequence of layers: *ReLU*, *Bilinear 2x2 Upsampler*, *Conv*, *BatchNorm*. Layer type *ConvTB* denotes the sequence: *Conv Transpose*, *ReLU*, and *BatchNorm*. Finally, layer type *ConvT* denotes the sequence: *Conv Transpose* and *ReLU*.

Encoder

Layer	Type	Stride / Kernel Size / Padding	Prev. Layer	Img. Size (pre layer)	Img. Size (post layer)
1	Conv	2/4/1	I_i	3x256x256	64x128x128
2	Conv	2/4/1	1	64x128x128	128x64x64
3	Conv	2/4/1	2	128x64x64	256x32x32
4	Conv	2/4/1	3	256x32x32	512x16x16
5	Conv	2/4/1	4	512x16x16	512x8x8
A	Conv	1/1	θ_i	2x1x1	32x1x1
B	Conv	1/1	A	32x1x1	32x1x1
6	Concat	–	5/B	–	544x8x8
7	Conv	2/4/1	6	544x8x8	512x4x4
8	Conv	2/4/1	7	512x4x4	512x2x2
9	Conv	2/4/1	8	512x2x2	512x1x1

Decoder (depth)

Layer	Type	Stride / Kernel Size / Padding	Prev. Layer	Img. Size (pre layer)	Img. Size (post layer)
10	UpSamp	1/3/1	x	512x1x1	512x2x2
11	UpSamp	1/3/1	10/8	1024x2x2	512x4x4
12	UpSamp	1/3/1	11/7	1024x4x4	512x8x8
13	UpSamp	1/3/1	12/5	1024x8x8	512x16x16
14	UpSamp	1/3/1	13/4	1024x16x16	256x32x32
15	UpSamp	1/3/1	14/3	512x32x32	128x64x64
16	UpSamp	1/3/1	15/2	256x64x64	64x128x128
17	UpSamp	1/3/1	16/1	128x128x12	3x256x256
18	Tanh (×5)	–	17	3x256x256	3x256x256

Decoder (silhouette) 256×256

Layer	Type	Stride / Kernel Size / Padding	Prev. Layer	Img. Size (pre layer)	Img. Size (post layer)
C	Conv	1/1	θ_i	2x1x1	32x1x1
D	Conv	1/1	A	32x1x1	32x1x1
19	Concat	–	D/x	–	544x1x1
20	ConvTB	2/4/1	19	544x1x1	256x4x4
21	ConvTB	2/4/1	20	256x4x4	128x8x8
22	ConvTB	2/4/1	21	128x8x8	128x16x16
23	ConvTB	2/4/1	22	128x16x16	64x32x32
24	ConvTB	2/4/1	23	64x32x32	64x64x64
25	ConvTB	2/4/1	24	64x64x64	32x128x128
26	ConvTB	2/4/1	25	32x128x128	1x256x256
27	Sigmoid	–	26	1x256x256	1x256x256

Decoder (silhouette) 57×57

Layer	Type	Stride / Kernel Size / Padding	Prev. Layer	Img. Size (pre layer)	Img. Size (post layer)
C	Conv	1/1	θ_i	2x1x1	32x1x1
D	Conv	1/1	A	32x1x1	32x1x1
19	Concat	–	D/x	–	544x1x1
20	ConvT	2/4/1	19	544x1x1	512x4x4
21	ConvT	2/4/1	20	512x4x4	256x8x8
22	ConvT	2/5/1	21	256x8x8	128x16x16
23	ConvT	2/5/1	22	128x16x16	64x32x32
24	ConvT	2/6/1	23	64x32x32	1x57x57
25	Sigmoid	–	24	1x57x57	1x57x57

References

Barron J, Malik J (2015) Shape, illumination, and reflectance from shading. *IEEE Transactions on Pattern Analysis and Machine Intelligence*

Besl P, McKay ND (1992) A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*

Blake A, Marinos C (1990) Shape from texture: estimation, isotropy and moments. Artificial Intelligence

Blanz V, Vetter T (1999) A morphable model for the synthesis of 3D faces. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics* pp 187–194

Blender Online Community (2017) Blender - a 3D modelling and rendering package. Blender Foundation, Blender Institute, Amsterdam

Boyer E, Franco J (2003) A hybrid approach for computing visual hulls of complex objects. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Cashman TJ, Fitzgibbon AW (2013) What shape are dolphins? building 3D morphable models from 2D images. *IEEE Transactions on Pattern Analysis and Machine Intelligence* pp 232–244

Chang A, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, et al (2015) Shapenet: An information-rich 3D model repository. arXiv preprint arXiv:151203012

Choy C, Xu D, Gwak J, Chen K, Savarese S (2016) 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. *Proceedings of the European Conference on Computer Vision*

Fan H, Su H, Guibas L (2016) A point set generation network for 3D object reconstruction from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Fouhey DF, Hussain W, Gupta A, Hebert M (2015) Single image 3D without a single 3D image. *Proceedings of the International Conference on Computer Vision*

Friedman J, Hastie T, Tibshirani R (2001) The elements of statistical learning, vol 1. Springer series in statistics New York

Gadelha M, Maji S, Wang R (2016) 3D shape induction from 2D views of multiple objects. arXiv preprint arXiv:161205872

Girdhar R, Fouhey D, Rodriguez M, Gupta A (2016) Learning a predictable and generative vector representation for objects. *Proceedings of the European Conference on Computer Vision* pp 484–499

Groueix T, Fisher M, Kim VG, Russell B, Aubry M (2018) Atlasnet: A papier-mâché approach to learning 3d surface generation. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Hartley RI, Zisserman A (2004) Multiple View Geometry in Computer Vision, 2nd edn. Cambridge University Press, ISBN: 0521540518

Hedau V, Hoiem D, Forsyth D (2009) Recovering the spatial layout of cluttered rooms. *Proceedings of the International Conference on Computer Vision*

Isola P, Zhu JY, Zhou T, Efros AA (2017) Image-to-image translation with conditional adversarial networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Jaderberg M, Simonyan K, Zisserman A, Kavukcuoglu K (2015) Spatial transformer networks. *Advances in Neural Information Processing Systems* pp 2017–2025

Kar A, Tulsiani S, Carreira J, Malik J (2015) Category-specific object reconstruction from a single image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Kar A, Häne C, Malik J (2017) Learning a multi-view stereo machine. *Advances in Neural Information Processing Systems* pp 364–375

Kolev K, Klodt M, Brox T, Cremers D (2009) Continuous global optimization in multiview 3D reconstruction. *International Journal of Computer Vision*

Kong C, Lin CH, Lucey S (2017) Using locally corresponding cad models for dense 3D reconstructions from a single image. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Laurentini A (1994) The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 16(2):150–162

Mahendran A, Vedaldi A (2015) Understanding deep image representations by inverting them. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Matusik W, Buehler C, Raskar R, Gortler S, McMillan L (2000) Image-based visual hulls. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*

Novotny D, Larlus D, Vedaldi A (2017) Learning 3D object categories by looking around them. *Proceedings of the International Conference on Computer Vision*

Park E, Yang J, Yumer E, Ceylan D, Berg A (2017) Transformation-grounded image generation network for novel 3D view synthesis. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Paszke A, Gross S, Chintala S, Chanan G, Yang E, DeVito Z, Lin Z, Desmaison A, Antiga L, Lerer A (2017) Automatic differentiation in pytorch. Proceedings of NIPS 2017 Workshop on Autodiff

Prasad M, Fitzgibbon AW, Zisserman A, Van Gool L (2010) Finding nemo: Deformable object class modelling using curve matching. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Qi CR, Su H, Niessner M, Dai A, Yan M, Guibas LJ (2016) Volumetric and multi-view cnns for object classification on 3D data. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Rezende D, Eslami SMA, Mohamed S, Battaglia P, Jaderberg M, Heess N (2016) Unsupervised learning of 3D structure from images. *Advances in Neural Information Processing Systems* pp 4997–5005

Rock J, Gupta T, Thorsen J, Gwak J, Shin D, Hoiem D (2015) Completing 3D object shape from one depth image. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Ronneberger O, Fischer P, Brox T (2015) U-net: Convolutional networks for biomedical image segmentation. In: *Proceedings of the International Conference on Medical Image Computing and Computer Assisted Intervention*

Seitz SM, Curless B, Diebel J, Scharstein D, Szeliski R (2006) A comparison and evaluation of multi-view stereo reconstruction algorithms. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* 1:519–528

Sinha A, Unmesh A, Huang Q, Ramani K (2017) Surfnet: Generating 3D shape surfaces using deep residual networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

SketchFab (2017) Sketchfab

Soltani AA, Huang H, Wu J, Kulkarni TD, Tenenbaum JB (2017) Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks.

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition

Su H, Maji S, Kalogerakis E, Learned-Miller E (2015) Multi-view convolutional neural networks for 3d shape recognition. In: *Proceedings of the International Conference on Computer Vision*

Tatarchenko M, Dosovitskiy A, Brox T (2016) Multi-view 3D models from single images with a convolutional network. *Proceedings of the European Conference on Computer Vision*

Tulsiani S, Zhou T, Efros A, Malik J (2017) Multi-view supervision for single-view reconstruction via differentiable ray consistency. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Upchurch P, Gardner J, Pleiss G, Pless R, Snavely N, Bala K, Weinberger K (2017) Deep feature interpolation for image content changes. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Vicente S, Carreira J, Agapito L, Batista J (2014) Reconstructing pascal voc. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*

Vogiatzis G, Torr PHS, Cipolla R (2003) Bayesian stochastic mesh optimization for 3D reconstruction. *Proceedings of the 14th British Machine Vision Conference, Norwich* pp 711–718

Wiles O, Zisserman A (2017) Silnet : Single- and multi-view reconstruction by learning from silhouettes. *Proceedings of the British Machine Vision Conference*

Witkin AP (1981) Recovering surface shape and orientation from texture. Artificial intelligence

Wu J, Zhang C, Xue T, Freeman B, Tenenbaum J (2016) Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. *Advances in Neural Information Processing Systems* pp 82–90

Wu J, Wang Y, Xue T, Sun X, Freeman B, Tenenbaum J (2017) Marrnet: 3D shape reconstruction via 2.5D sketches. *Advances in Neural Information Processing Systems*

Xiang Y, Mottaghi R, Savarese S (2014) Beyond pascal: A benchmark for 3d object detection in the wild. In: *Proceedings of the IEEE Workshop on Applications of Computer Vision*

Yan X, Yang J, Yumer E, Guo Y, Lee H (2016) Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision. *Advances in Neural Information Processing Systems*

Zhang R, Tsai PS, Cryer JE, Shah M (1999) Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 21(8):690–706

Zhou QY, Park J, Koltun V (2018) Open3D: A modern library for 3D data processing. arXiv:180109847

Zhou T, Tulsiani S, Sun W, Malik J, Efros A (2016) View synthesis by appearance flow. *Proceedings of the European Conference on Computer Vision*

Zhu R, Galoogahi HK, Wang C, Lucey S (2017) Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. *Proceedings of the International Conference on Computer Vision* pp 57–65

Zollhöfer M, Dai A, Innmann M, Wu C, Stamminger M, Theobalt C, Nießner M (2015) Shading-based refinement on volumetric signed distance functions. *ACM Transactions on Graphics (TOG)*

Chapter 5

3D Surface Reconstruction by Pointillism

This work was presented as an oral and published at the ECCV Workshop on *Geometry meets Deep Learning*.

This chapter describes an approach for self-supervised single image depth prediction using correspondences extracted from multi-view geometry, as discussed in Section 1.2. Using correspondences allows the method to be applied to images that vary significantly in illumination, context, and viewpoint. In these cases a standard photometric self-supervised loss would not work.

3D Surface Reconstruction by Pointillism

Olivia Wiles and Andrew Zisserman

Visual Geometry Group,
University of Oxford
{ow,az}@robots.ox.ac.uk

Abstract. The objective of this work is to infer the 3D shape of an object from a single image. We use sculptures as our training and test bed, as these have great variety in shape and appearance.
To achieve this we build on the success of multiple view geometry which is able to accurately provide *correspondences* between images of 3D objects under varying viewpoint and illumination conditions, and make the following contributions: first, we introduce a new loss function that can harness image-to-image correspondences to provide a supervisory signal to train a deep network to infer a depth map. The network is trained end-to-end by differentiating through the camera. Second, we develop a processing pipeline to automatically generate a very large scale multi-view set of correspondences for training the network. Finally, we demonstrate that we can indeed obtain a depth map of a novel object from a single image for a variety of sculptures with varying shape and texture, and that the network generalises at test time to new domains, such as synthetic data.

1 Introduction

Humans are able to effortlessly perceive 3D shape of a previously unseen object from a single image – or at least we have the impression that we do this. For example for a piecewise smooth sculpture such as the one by Henry Moore in Fig. 1, we know where there are concavities, convexities and saddles, as well as where there are holes and sharp points. *How* this is achieved has long been studied in computer vision in terms of geometric cues from the silhouette [1], the texture [2–4], self-shadows, specularities [5], shading [6, 7], chiaroscuro [8], etc.

In this paper our objective is to be able to reconstruct such objects from a single image. Deep learning has significantly boosted progress in 3D reconstruction from single images, but so far methods have mostly depended on the availability of synthetic 3D training examples, or using a single class, or pre-processing the data using SfM and MVS to extract depth. In contrast, our self-supervised approach is to learn directly from real images, capitalizing on many years of research on multiple view geometry [9–12] that is able to automatically determine matching views of a 3D object and generate point correspondences, without requiring any explicit 3D information as supervision.

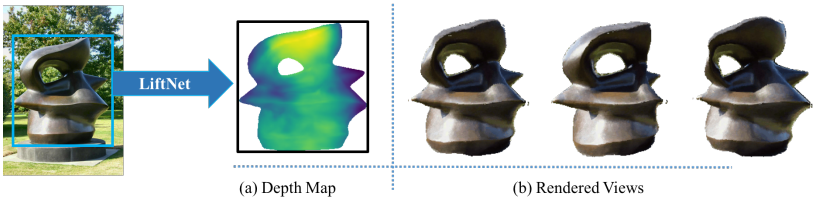


Fig. 1: Given this image of the *Spindle Piece* by *Henry Moore*, LiftNet predicts the 3D shape of the sculpture via a depth map (a), where blue is further away and yellow nearer, and thereby enables new views to be rendered (b). LiftNet is trained self-supervised on real images using correspondences *without* any knowledge of depth or camera parameters.

The key idea is to use image-to-image point correspondences to provide a *training* loss on the depth map predicted by a Convolutional Neural Network (CNN), called *LiftNet*. This is illustrated in figure 2. Suppose we are attempting to infer the depth of the object in a *source* view, and there are a number of image point correspondences available between the *source* view and a *target* view (where a correspondence is defined by the projection of a 3D surface point into the source and target views). A correspondence can be computed in two ways. First, it can be computed using matching methods from multiple view geometry (such as SIFT, and epipolar geometry). This method does not involve using the depth of the surface and we treat these correspondences as ground truth. Second, it can be computed by inferring the depth of the point in the *source* view and projecting the 3D point into the *target* view. If the CNN correctly predicts the depth of the points in the *source* view, then the projected points will coincide with the ground truth correspondences in the *target* view; however, if the the depth prediction is incorrect, then the distance between the projected and corresponding points – the re-projection error – defines a loss that can be used to train the network.

Of course, the correspondences between two views of a particular sculpture only provide constraints at those points on the surface – and correspondences will mainly be found at surface texture, surface discontinuities, and boundaries [13], i.e. not uniformly across the surface. However, for each sculpture there are multiple pairs of images; and each pair can 'probe' (and constrain) different points on the surface according to its correspondences. Finally, and most importantly, the network must learn to predict correspondences not just for a particular sculpture, but for *all* the sculptures (and all their view pairs) in the training set – and we have 170K training pairs and around 31M training correspondences. The only way it can solve this task is to infer 3D shape for each image.

To this end, we formulate a new deep learning framework for extracting 3D shape which is similar to the artistic *pointillist* style. Analogously to how pointillists build up colour variation in an image from dots of discrete colour, we use points in correspondence between images of an object in order to train a network over time to learn the 3D shape of the object.

Contributions. This work presents three contributions: first, to use corresponding points to formulate a differentiable loss on the object shape that can be used to train a network from scratch (Section 3). The formulation includes differentiating through the camera to train the network end-to-end.

The second contribution is a pipeline based on multiple view geometry for automatically extracting robust correspondences between multiple pairs of images of a sculpture (Section 4). We use these correspondences to train the network on real images, *without* ground truth 3D depth information. This is done entirely automatically and is the first system to our knowledge to learn to predict shape end-to-end for a set of objects by using correspondences and geometry in this manner.

The final contribution is our experimental results in Section 5, which demonstrate that the trained network can not only predict depth for the given domain but also generalises to synthetic data, allowing its generalisation capability to be evaluated quantitatively.

2 Related Work

Depth Prediction. The ability to learn depth using a deep learning framework was introduced by [14], who use a dataset of ground truth depth and RGB image pairs to train a network to predict depth. This has been improved on with better architectures in [15, 16] and generalised to ordinal relationships in [17, 18].

A recent set of works have considered how to extract the 3D depth of a scene between pairs of images without knowing the camera motion or depth [19–22]. This is done by predicting both depth and cameras in the network. This information is then used to transform one view and the photometric error between the generated image and the ground truth is used to train the network. These works require that the two images be very similar, such that the photometric error gives a robust and sensible loss. As a result, the images come from stereo datasets or consecutive video frames, such that the relative appearance change is small. On the other hand, our approach uses point correspondences directly, and consequently the images can vary dramatically in illumination, texture, size, position, etc. and our loss is robust to these changes.

3D Shape Prediction. Going beyond depth prediction, which is view based, the entire 3D shape of the object can be reconstructed from multiple views by using strong supervision from the known 3D geometry to predict a voxel [23, 24] or point cloud [25–27] representation. Alternatively, the supervision can be from photo consistency or silhouette constraints [28–32]. However, these methods require knowledge of the camera parameters in order to enforce the geometric constraints.

These methods have been extended to deal with natural images in the work of [33–35], but [33] still requires a synthetic dataset on which to train their network which is then fine-tuned on real images. [34] uses structure from motion (SfM)/multi-view stereo (MVS) [9] from a video sequence as the ground truth 3D shape on which to train their network for reconstructing a finite set of classes;

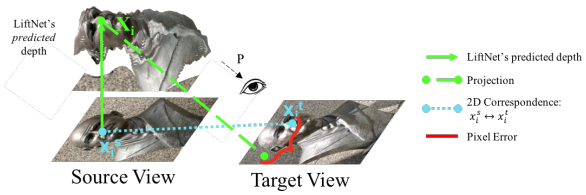


Fig. 2: An illustration of the training loss: $\mathcal{L}_{\text{corr}}$. Given $\mathbf{x}_i^s \leftrightarrow \mathbf{x}_i^t$ and the best camera $\mathtt{P}$, we minimise the error between $\mathtt{P}\mathbf{X}_i$ and $\mathbf{x}_i^t$. The depth d_i value of $\mathbf{X}_i$ is LiftNet's predicted depth for the *source* view at $\mathbf{x}_i^s$. If d_i were correctly predicted by LiftNet there would be no error as $\mathbf{X}_i$ would project to $\mathbf{x}_i^t$; the image distance between the projected point and $\mathbf{x}_i^t$ provides the training loss. As the network's prediction improves, the distance reduces to zero.

[35] extends this idea to unordered image collection of historic landmarks by using many images of the given landmark. In our case, we are not restricted to a finite set of classes, and do not require a video sequence or many images of the same scene in order to obtain a dense reconstruction, but instead train from the available correspondences directly, and these correspondences only need to exist over a handful of images. As a result, our approach can be used with far fewer samples of each landmark or sculpture.

3 Approach

The goal is to recover 3D structure from a single image by predicting a depth map, but *without* requiring ground truth 3D information in training. In this section we first define the loss functions used to train the network. Then the LiftNet architecture is described in Section 3.3. In the following we assume that correspondences between images are available (as described in Section 4).

As introduced in Section 1, the depth predicted by the LiftNet CNN in the *source* view is supervised by using point correspondences as follows: (i) let the set of correspondences be denoted as $\mathbf{x}_i^s \leftrightarrow \mathbf{x}_i^t$, where $\mathbf{x}_i^s$ are the points in the *source* view, and $\mathbf{x}_i^t$ the points in the *target* view. (ii) Then in the *source* view we can determine the 3D points $\mathbf{X}_i$ that project to $\mathbf{x}_i^s$ (since the network gives the depth of each point). (iii) Since we know the correspondence between $\mathbf{X}_i$ and $\mathbf{x}_i^t$ we can compute the best camera that projects the 3D points $\mathbf{X}_i$ into the *target* view. (iv) If the 3D shape has been predicted perfectly, then the 3D points $\mathbf{X}_i$ will project perfectly onto $\mathbf{x}_i^t$. If they do not, then this *reprojection error* provides a loss that can be minimized to train the network. The resulting loss is defined as:

$$\mathcal{L}_{\text{corr}} = \frac{1}{N}\sum_{i=1}^{N} d_R(\mathtt{P}\mathbf{X}_i, \mathbf{x}_i^t); \tag{1}$$

where $d_R(.,.)$ denotes the Euclidean (L_2) pixel distance between vectors subject to a robustness function R.

This loss is a useful constraint, as it enforces important properties of the object, such as concavities and convexities. Moreover, this can be done for any pair of images for which correspondences can be obtained. There is no requirement that the images be photometrically consistent – e.g. lighting, texture, position etc. can vary dramatically between views.

Finally, a robustness term R is added (Section 3.2), as the 2D correspondences may be noisy (as explained in Section 4).

3.1 Point Correspondence Loss $\mathcal{L}_{\mathbf{corr}}$

We minimise the projection error between $\mathbf{X}_i$ and $\mathbf{x}_i^t$ using the best camera $\mathtt{P}$: $\frac{1}{N}\sum_{i=1}^{N} d_R(\mathtt{P}\mathbf{X}_i, \mathbf{x}_i^t)$. The steps are as follows:

A. Choose the camera. This work assumes an affine camera and an orthogonal coordinate system in the *source* view, which is why $\mathbf{X}_i = [x_i^s, y_i^s, d_i, 1]^T$ projects to $\mathbf{x}_i^s = (x_i^s, y_i^s)$ in the *source* view. As has been noted previously [9, 36, 37], the affine case is a very stable and useful approximation to perspective projection. The reader is referred to the supplementary material for a detailed review of this camera model. However, we note that the ideas presented here (e.g. the method of differentiating the camera) generalise in a straight forward manner to the perspective case.

B. Determine the camera. We first determine the camera matrix $\mathtt{P}$ by solving the system of equations $\mathbf{x}_i^t = \mathtt{P}\ \mathbf{X}_i$ for $\mathtt{P}$. We know which values $\mathbf{X}_i$ and $\mathbf{x}_i^t$ should correspond because LiftNet's prediction is simply a depth map, so $\mathbf{x}_i^s = (x_i^s, y_i^s)$ in pixels maps to $\mathbf{X}_i = [x_i^s, y_i^s, d_i, 1]^T$ (d_i is the depth prediction at that point) and we know, via the correspondences, that $\mathbf{x}_i^s \leftrightarrow \mathbf{x}_i^t$, so $\mathbf{X}_i$ maps to $\mathbf{x}_i^t$. This gives the following system of equations:

$$\begin{bmatrix} x_1^t & \dots & x_N^t \\ y_1^t & \dots & y_N^t \end{bmatrix} = \mathtt{P} \begin{bmatrix} x_1^s & \dots & x_N^s \\ y_1^s & \dots & y_N^s \\ d_1 & \dots & d_N \\ 1 & \dots & 1 \end{bmatrix}. \tag{2}$$

However, directly solving the system of equations would be problematic due to the effect of outliers, (e.g. noise in the data). A standard approach to deal with noise is to make use of RANSAC [38]. This method solves a system of equations by finding a solution that satisfies the most constraints. The satisfied constraints are called inliers, the others outliers. In our case, we want to find $\mathtt{P}$ such that the maximum number of pairs $\mathbf{X}_i$ and $\mathbf{x}_i^t$ satisfy the condition $|\mathbf{x}_i^t - \mathtt{P}\mathbf{X}_i|_2 < T$ for some threshold T. Given the set of inliers $\mathbf{X}_{i_{\text{inliers}}}, \mathbf{x}_{i_{\text{inliers}}}^t$, a new system of linear equations is constructed: $\mathbf{x}_{i_{\text{inliers}}}^t = \mathtt{P}\mathbf{X}_{i_{\text{inliers}}}$.

C. Compute the loss. Given $\mathtt{P}$, all points $\mathbf{X}_i$ are projected into the *target* view and the error between their projection and known location $\mathbf{x}_i^t$ is computed. The loss is then as given in (1).

D. Differentiate through the camera. In order to train the network end to end, it is necessary to compute the derivative $\frac{\partial \mathtt{P}}{\partial d_i}$. To do this, we re-write the system of equations such that $\mathtt{P}$ is explicitly a function of $x^t_{i_{\text{inliers}}}/\mathbf{X}_{i_{\text{inliers}}}$ such that computing the derivative is straightforward. For ease of notation, the matrix of inliers $\mathbf{X}_{i_{\text{inliers}}}$ is referred to as $\mathtt{X}$ and of inliers $\mathtt{x}_{i_{\text{inliers}}}$ as $\mathtt{x}$ from now on. The pseudo-inverse $\mathtt{X}^+$ is computed using the singular value decomposition (SVD) [39]. (If the SVD of a matrix $\mathtt{A}$ is $\mathtt{A} = \mathtt{U}\Sigma\mathtt{V}^T$ then its pseudo inverse can be written as $\mathtt{A}^+ = \mathtt{V}\Sigma^+\mathtt{U}^T$.) Then the system of equations can be re-written as:

$$\mathtt{x} = \mathtt{PX} \tag{3}$$

$$\mathtt{x}^T = \mathtt{X}^T\mathtt{P}^T \tag{4}$$

$$(\mathtt{X}^T)^+\mathtt{x}^T \approx (\mathtt{X}^T)^+\mathtt{X}^T\mathtt{P}^T \approx \mathtt{P}^T \tag{5}$$

$$(\mathtt{V}\Sigma^+\mathtt{U}^T)\mathtt{x}^T = \mathtt{P}^T \tag{6}$$

Note that because the system of equations in (2) is over-constrained then of course this is not an exact solution, but the pseudo-inverse solves the system of equations in the least-squares sense [39] which is what we require. Also, note that because the outliers are ignored when computing $\mathtt{P}$ due to RANSAC, we can ignore them in this computation and only consider the inliers. This gives the forward pass.

To perform the backward pass, it is necessary to compute the derivative $\frac{\partial \mathtt{P}}{\partial d_i}$. This is a straight-forward application of the product and chain rule, except for the computation of the SVD. However, previous work (e.g. [40]) has demonstrated how to compute these derivatives. As a result we can back-propagate through the computation of $\mathtt{P}$ to the estimation of the height values d_i. This is achieved in practice using standard layers in a neural network library (e.g. PyTorch [41]). Note that computing the gradients for $\mathtt{U}$, $\mathtt{V}$ could lead to potential instability if $\mathtt{X}^T$ is not full rank or has repeated singular values; however, this was not a problem in practice.

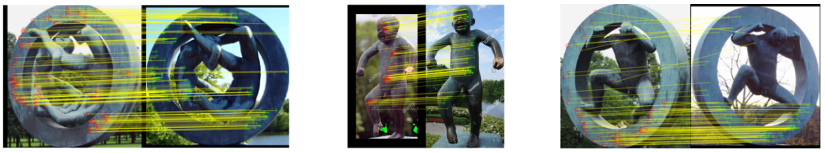
Discussion. We note that our method computes $\mathbf{d}$ up to an overall affine ambiguity. This amounts to a scaling and shearing in the depth prediction. This ambiguity is seen in human vision, as humans have been shown to reconstruct objects (such as vases) up to an affine transformation in depth [42, 43]. It is hypothesised that this difficulty arises from the fact that, assuming Lambertian reflectance and given a single image, the surface of an object can only be recovered up to a generalized bas-relief ambiguity [44].

3.2 Robustness

As the correspondences and segmentations will be noisy, it is necessary that the loss function is robust to these errors. To do this we use a smooth function to weight the errors [45] so that errors above a threshold τ are given a constant cost: $\mathcal{R}(x) = \begin{cases} \frac{1}{2}x^2(1 - \frac{x^2}{2\tau^2}), & \text{if } x^2 \leq \tau^2 \\ \tau^2/4, & \text{otherwise} \end{cases}$.



(a) The images are organised by artist (solid lines) into clusters (dashed lines). Note the variety and complexity of the sculptures: in shape, materials, lighting, and viewpoint.



(b) Sample correspondences for pairs of images. The images may be taken at different times of year, in different contexts/illumination conditions and the material itself may change over time due to weather.

Fig. 3: **The Sculpture dataset.** Note that this is only a tiny subset of the clusters and a fraction of the number of images within a cluster. Please refer to the supp. material for more examples.

3.3 Architecture

The architecture used is based on the U-Net [46] variant of pix2pix [47]. This architecture includes skip connections in order to maintain high level detail. However, we incorporate two modifications. First the last activation is replaced by a tanh layer to enforce that the output lies between $[-1, 1]$. We impose this range so that the predicted depth does not grow too large, making training unstable. As LiftNet learns depth up to a scaling factor in depth, this in no way constrains the types of surfaces that LiftNet can describe. Second, the nearest neighbour upsampler is replaced by a bilinear upsampler. This mitigates against pixelated effects [48, 49]. Please refer to the supplementary material for full details on the architecture.

4 The Sculpture Dataset

We assemble a large scale dataset of images of sculptures for training and testing by combining multiple public datasets [50, 13, 51, 52] and downloading additional images from the web. The dataset incorporates a wide variety of artists, styles and materials. It is divided at the artist level to prevent any information bleeding between the sets. Table 1 gives the number of artists and works used as well as the train/val/test splits.

The dataset includes multiple works (sculptures) by different artists (sculptors) organised into a set of clusters. Within a cluster, the images are of the same

Table 1: Dataset statistics for the Sculpture dataset. Note the large number of artists and works that are used. This results in a large variety of styles and shapes that LiftNet must contend with. A total of 31M point correspondences ($\approx$ 181 correspondences $\times$ $169K$ pairs) are automatically generated and used to train LiftNet.

	Train	Val	Test	All
#Artists	138	7	1	143
#Works	1031	27	129	1187
#Matching Pairs	168726	552	13166	182K
Avg # Correspondences per Pair	181	223	174	181

sculpture (shape), but there may be multiple instances of the sculpture, some made of different material. The utility of the dataset is that within a cluster there are many point correspondences between image pairs that can be used for training the network. Fig. 3 shows a sample of sculptures, correspondences and an example cluster.

The remainder of this section describes the steps used to download, prepare, and obtain the image pair correspondences of the dataset. Additional details are given in the supplementary material.

Image extraction. We combine multiple sculpture datasets: [50, 13, 51, 52] and download additional images from the web.

Obtaining segmentations. To segment the images, RefineNET [53] is trained on 2000 hand-labelled sculptures by artists *Rodin* and *Henry Moore*. It achieves a 0.94 IoU score and 0.97 accuracy on the validation dataset. This is used for a wide variety of images and it generalises well to new sculptures.

Obtaining correspondences. The final step is to determine a valid set of correspondences. The OpenMVG pipeline [54] is used to extract an initial dense list of correspondences between pairs of images. The segmentation from RefineNET above is then used to mask out correspondences from the irrelevant background parts of the image. Additionally those correspondences that do not satisfy the affine fundamental matrix, which is computed using RANSAC, are removed. Finally, those image pairs that can be mapped by an affine homography (i.e. a 2D transformation between images) are thrown out, as they will not provide a constraint on 3D structure.

Despite these post-processing steps, there will still be noise in the correspondences, motivating the use of a robust cost in our losses explained in Section 3.

5 Experiments

A challenge of our framework is to determine its prediction quality, as there is *no* ground truth depth information for the automatically collected Sculpture dataset. To this end, LiftNet is evaluated in multiple environments and scenarios. *First*, we use a realistic synthetic dataset of sculptures SketchFab [55] and ShapeNet [56] for which we can determine ground truth information and thereby correspondences between views; these are introduced below. LiftNet is

then trained using these generated correspondences and compared to a baseline trained to explicitly regress depth on Section 5.3 and Section 5.4. *Second*, we train LiftNet on real data: the Sculpture dataset. This network is then compared to a number of self-supervised and supervised methods in Section 5.5. This evaluation is performed on two datasets: first it is performed on Scanned, a dataset of scanned objects. Second, the evaluation is performed on SketchFab (despite the domain gap between real and synthetic images the network generalises to this new domain). Finally, it is evaluated qualitatively on the Sculpture dataset in Section 5.6.

5.1 Datasets, evaluation metrics, and baselines

The SketchFab and ShapeNet datasets. SketchFab is a large dataset of synthetic 3D models of sculptures generated using photogrammetry. There are 425 sculptures divided into 372/20/33 train/val/test sculptures. ShapeNet consists of multiple semantic classes, each of which is divided into train/val/test using the given splits. For evaluation, five views of each SketchFab object and 10 views of each ShapeNet object are rendered in Blender [57] using orthographic projection and the ground truth depth extracted. The SketchFab objects are viewed with azimuth $\in [0°, 120°]$, elevation $0°$ whereas ShapeNet objects are viewed with azimuth $\in [0°, 360°]$ and elevation $\in [-45°, 45°]$. As the depth and cameras of the renders are known, then the ground truth correspondences between images can be determined by projecting the depth in the *source* view into the *target* view.

Scanned. Additional data is collected from the 80 sculpture videos of [58]. These are taken 'in-the-wild' with a hand-held camera. Of these videos, 11 objects are chosen and the sculpture region segmented. This gives 208 images for testing.

Evaluation metrics. The results are reported using multiple metrics: the L_1 error, root mean squared error, relative L_1 error, and squared rel. difference [14]. To evaluate the depth prediction, it is necessary to take into account the ambiguity in the z axis (the depth prediction). This is done by allowing for a scaling/translation in depth. Thus for all models (including those trained on ground truth depth), when reporting results, the depth prediction d_{pred} for an image is first normalised by $d^*_{\text{pred}} = \alpha(d_{\text{pred}} - \beta_1) + \beta_2$ where β_1 is the median of d_{pred}, β_2 is the median of d_{gt} and α allows for a scaling in depth: $\alpha = \sum_{xy}(d_{\text{pred}} * d_{\text{gt}}) / \sum_{xy}(d^2_{\text{pred}})$. ($d_{gt}$ denotes ground truth and $\sum_{xy}$ denotes summation over pixel locations.)

Baselines. In the evaluation on synthetic data, we compare against a supervised baseline, explicitly trained to regress depth. We use the same network (e.g. pix2pix) as LiftNet. The MSELoss is used but after first accounting for a scaling and translation in depth as follows. If the depth predicted is d_{pred} then the normalised depth is $d^*_{\text{pred}} = \alpha(d_{\text{pred}} - \beta_1) + \beta_2$, which is computed as described above for the evaluation metrics. The loss is then $|d^*_{\text{pred}} - d_{gt}|_2$.

5.2 Training

The network is trained as follows. Two images with correspondences are sampled from the dataset; one is designated *source*, the other *target*. The *source* view is then input to LiftNet, which predicts the depth at all pixels. The predicted depth of the foreground pixels d_i, concatenated with the x_i,y_i position of the pixel in the image give the 3D points in the *source* view $\mathbf{X}_i = [x_i, y_i, d_i, 1]^T$. The correspondence loss – $\mathcal{L}_{\text{corr}}$ – is then imposed on these 3D points.

At *test* time (visualised in Fig. 4), an image is simply input to the network. This gives the depth prediction for all pixels. For visualisation purposes, the sculpture (the foreground pixels) are segmented from the background and only the depth values at these foreground pixels is shown.

The models are trained on a single Titan GPU in PyTorch [59]. They take about half a day to train. All models trained on the Sculpture dataset are trained

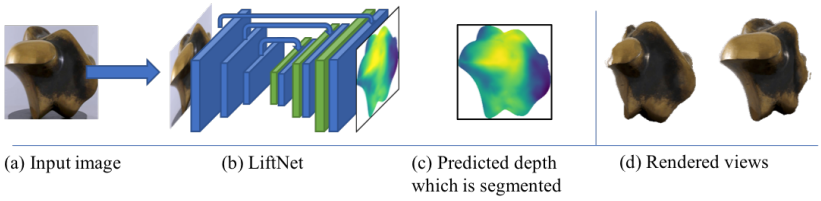


Fig. 4: The test time pipeline for LiftNet. (a) An image is selected from the test dataset and input to LiftNet (Section 3.3). (b) LiftNet gives a depth map prediction at all points. (c-d) The rendered depth is then segmented and visualised at new views. (This is a sample result on the test set.)

as follows. The models are trained with SGD, a learning rate of $1e^{-5}$, and momentum of 0.9. The gradients are clamped to ± 5. These models are trained until the correspondence error on the Sculpture dataset's validation set stops decreasing. When trained on SketchFab or ShapeNet, models are trained with SGD a learning rate of $1e^{-3}$, and momentum of 0.9. The gradients are clamped to ± 5. They are trained until the correspondence error on the validation set stops decreasing or a maximum of 200 epochs.

5.3 Quantitative results on ShapeNet

In this section, we evaluate LiftNet on ShapeNet. In order to test the correspondence loss, 50 correspondences per pair of images of an object are randomly chosen and fixed using the known depth and camera transformation. This gives the training set.

The results are reported in Table 2 and LiftNet is compared to training the same network architecture (i.e. pix2pix) but directly regressing the ground truth depth up to a scaling and translation in depth as described above.

Table 2: Comparison of 'LiftNet trained on ShapeNet correspondences' to 'pix2pix trained using a MSE loss on ShapeNet'. For both methods, the error measure is RMSE (x100).

	rif.	boo.	bus	bed	spe.	cab.	lam.	cha.	tra.	pla.	tab.	dis.	mot.	car	wat.	pho.	sofa
pix2pix	**1.71**	2.14	**1.89**	2.16	1.66	1.76	1.44	1.87	**1.71**	**0.90**	2.52	2.28	**1.71**	1.33	**1.36**	1.56	2.19
LiftNet	2.03	**1.94**	2.11	**1.21**	**1.29**	**1.21**	**1.38**	**0.94**	2.06	1.05	**1.51**	**1.66**	1.92	**1.12**	1.51	**1.52**	**1.34**

These results are perhaps surprising, as LiftNet does better on multiple classes and comparably on most. Thus, training with a limited number of correspondences can yield comparable results to training with dense depth.

5.4 Quantitative results on SketchFab

In this section, LiftNet is evaluated on a synthetic dataset of sculptures, SketchFab, which has more varied shapes than ShapeNet. LiftNet is trained using ground truth correspondences for SketchFab for every pixel (i.e. dense points). LiftNet's performance is then compared with the baseline methods trained with depth. As demonstrated in Table 4, our method performs similarly to the supervised method trained explicitly to regress depth. Qualitative results are given in the supplementary material.

While here we have used all points, for ShapeNet only 50 correspondences was sufficient. Consequently, we additionally investigate in Table 3 the performance as a function of the number of training correspondences used per image and demonstrate that using a fraction of the available number of correspondences gives comparable results to using all. For example, using 100 correspondences gives similar results – 0.175/0.254 L1/RMSE error versus 0.175/0.255; we can use 1.1% of the correspondences and achieve comparable results to using all.

Table 3: Evaluation of LiftNet's robustness to the number of training correspondences. Lower is better. These results demonstrate that using only 50 correspondences per training pair gives similar results to using all. Thus, sparse correspondences are sufficient for training LiftNet.

# Correspondences per Image	L_1	RMSE	$\frac{d^*-d_{gt}}{d_{gt}}$	$\frac{(d^*-d_{gt})^2}{d_{gt}}$
10	0.183	0.263	0.0673	0.0253
50	0.178	0.261	0.0650	0.0242
100	0.175	0.254	0.0640	0.0233
≈9000	0.175	0.255	0.0641	0.0233

5.5 Quantitative results using real world data

Given the initial experiments on ShapeNet and SketchFab, which demonstrate that our loss is sufficient to learn about 3D and that using sparse correspondences is powerful, we turn our attention to using real-world, noisy data. However, as there is no large dataset of ground truth 3D sculptures, we evaluate on two

datasets. First, we evaluate on real images using the Scanned dataset. Second we evaluate the models' generalisation capabilities by evaluating on SketchFab. To perform well, these models must generalise to a new, synthetic domain which may require a challenging domain shift. However, in practice, the renderings are high quality enough and the model robust enough that this does not seem to be a problem.

Training. When training, the loss on the validation set decreases from ≈ 4.0 to ≈ 3.4, converging in $40K$ iterations.

Table 4: The performance of LiftNet evaluated on the SketchFab dataset. Across all metrics, lower is better.

Method	Trained with	Training Dataset	L_1	RMSE	$\frac{d^*-d_{gt}}{d_{gt}}$	$\frac{(d^*-d_{gt})^2}{d_{gt}}$
COLMAP [60]	Depth from SfM	Sculptures	0.195	0.284	0.0760	0.0291
LiftNet: $\mathcal{L}_{\text{Corr}}$ (no R)	Correspondences	Sculptures	0.190	0.277	0.0690	0.0269
LiftNet: $\mathcal{L}_{\text{Corr}}$	Correspondences	Sculptures	**0.186**	**0.270**	**0.0677**	**0.0256**
Zhou et. al. [21]	Photometric Error	Sculptures	0.202	0.291	0.0732	0.0297
Chen *et al.* [17]	Ground Truth Ordinal Depth	Depth-in-the-Wild	0.186	0.269	0.0680	0.0258
LiftNet: $\mathcal{L}_{\text{Corr}}$	Correspondences	SketchFab	0.175	**0.254**	0.0641	0.0233
pix2pix	Depth	SketchFab	**0.173**	**0.254**	**0.0628**	**0.0226**

Table 5: The performance of LiftNet evaluated on the Scanned dataset. Across all metrics, lower is better.

Method	Trained with	Training Dataset	$L_1(cm)$	RMSE (cm)	$\frac{d^*-d_{gt}}{d_{gt}}$	$\frac{(d^*-d_{gt})^2}{d_{gt}}$
COLMAP [60]	Depth from SfM	Sculptures	9.5	11.8	**0.0741**	18.1
LiftNet: $\mathcal{L}_{\text{Corr}}$	Correspondences	Sculptures	**9.4**	**11.6**	**0.0741**	**16.6**
Zhou et. al. [21]	Photometric Error	Sculptures	9.8	12.1	0.0761	18.7
Chen *et al.* [17]	Ground Truth Ordinal Depth	Depth-in-the-Wild	9.3	11.7	0.0722	17.1

Ablation Studies. The first step is to ensure that our loss does indeed enforce that LiftNet learns about depth. To perform this check, we evaluate LiftNet on the test set of SketchFab and evaluate the effect of adding each component: the correspondence loss $\mathcal{L}_{\text{corr}}$ and the robustness term R.

The results are reported in Table 4. From these results, it is clear that the correspondence loss provides a strong constraint on the predicted depth, which is improved by the robustness term.

Comparison to SfM. The benefit of our approach is that we do not require videos of the same object but instead can use unordered image collections and a small number of images per object. To demonstrate this, we compare to COLMAP [60]. COLMAP is run on the clusters and the recovered 3D used to train a model to explicitly regress depth. COLMAP failed for 77% of the clusters, as there are not sufficient images/correspondences for it to converge to a global solution. Table 4 and 5 compares the performance of the two methods. The proposed pipeline and LiftNet training are superior, due to (we assume):

(1) more data for training, as no correspondences are thrown out; and (2) that the depth from COLMAP may be incorrect due to the small number of images per cluster, which may lead to an incorrect solution.

This experiment suggests that our method is additionally useful when fine-tuning a pre-trained model (e.g. with ground truth depth) on a new domain with only a few images per instance (e.g. lesser known landmarks, sculptures, etc.) as a SfM approach would fail given the sparse amount of information.

Comparison to other self-supervised approaches. The second hypothesis to test is whether our method is more robust than other self-supervised methods which rely on photometric consistency. We compare to the work of [21] by running their model on our dataset. However, we note that their method requires knowledge of the intrinsic camera parameters which we do not have. As a result, we assume the intrinsic camera parameters have focal length 0.7*W, and the principal point is (0.5W, 0.5H) (W/H are the width/height of the image). The results are reported in Table 4. As can be seen their model does poorly: this is presumably due to a number of challenging characteristics of the Sculpture dataset. First, as mentioned above the intrinsic camera parameters are not known and may change from image to image. Second, there are large changes in illumination, changes in context, changes in weather, etc. All of these characteristics make using a photometric loss not robust and lead to worse results.

Comparison to supervised approaches. Despite LiftNet doing better than comparable self-supervised approaches, as reported above, the next question is how does LiftNet compare to a method [17] trained with depth supervision. [17] is pre-trained on the NYU depth dataset [61] which contains 795 densely annotated images and fine-tuned on the depth-in-the-wild [17] which contains 5M images with ordinal relationships. As demonstrated in Table 4 and 5, LiftNet does comparably or better than this *supervised* baseline.

5.6 Qualitative Results on the Sculpture dataset

We demonstrate in Fig. 5 the predictions of LiftNet on the testing portion of the Sculpture dataset and compare them visually to two other methods: COLMAP and the supervised method [17]. We note that COLMAP performs poorly, presumably as there are very few training points. [17] produces reasonable results, as it is trained on a large dataset of outdoors images with supervision on relative depth in addition to NYU, but it has certain priors over the image (e.g. that points in the bottom of the image are always nearer than those in the top – as for most images the foreground is at the bottom of the image and sky at the top). Please see the supplementary material for more results.

6 Discussion

In this paper, we have introduced a framework for learning 3D shape using easily attainable sparse correspondences *without* depth supervision. Our insight is that we can make use of sparse correspondences, which can be obtained in much less

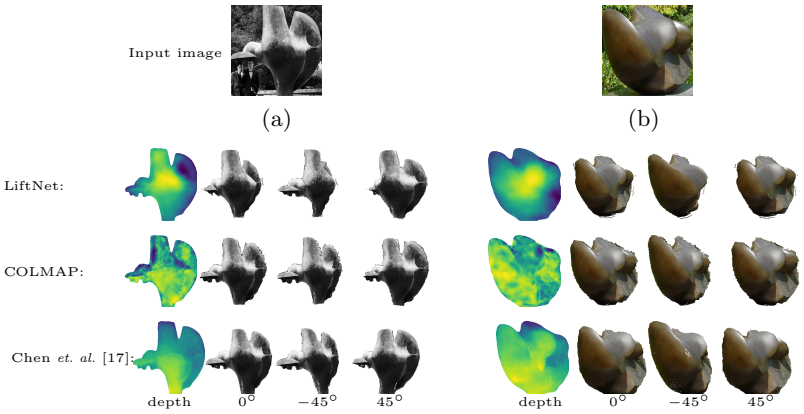


Fig. 5: Reconstruction results for LiftNet (top), COLMAP (middle) and Chen *et al.* [17] (bottom), visualised using Open3D [62]. The input image is shown at the top, then the predicted depth (blue is further away, yellow nearer), and rendered 3D at multiple viewpoints. Zoom in for details. From these images, the following are demonstrated. First, Chen *et al.* learns a prior over the image that the bottom of the image is nearer and the top further away. This is demonstrated in (a) and further examples in the supp. material. Second, COLMAP's depth predictions are noisy. Finally, LiftNet produces convincing depth maps which can be rendered at new views.

constrained environments than approaches requiring photometric consistency. Given enough sparse correspondences across many instances, the network learns a dense depth prediction. The approach has been demonstrated on a challenging sculpture dataset of real images and a synthetic sculpture dataset with known ground truth information.

It is interesting to consider why this training scenario based on real images, and sculptures in particular, produces a network that performs well on real images and also generalizes to synthetic image. It is probably in part because the training data has natural augmentation – instances of a sculpture with the same shape may be made from different materials (bronze, marble) or have different texturing and appearance due to different weathering or illumination conditions. The network must learn to produce the same shape, irrespective of these multifarious conditions. This is a challenging learning problem but, if successful, then the network has correctly learnt to disentangle the material/appearance from the shape, and to pick out cues to shape from appearance. Thus it can generalize to objects with different materials, e.g. synthetic ones.

Acknowledgements. The authors would like to thank Fatma Guney for helpful feedback and suggestions. This work was funded by an EPSRC studentship and EPSRC Programme Grant Seebibyte EP/M013774/1.

References

1. Koenderink, J.J.: What does the occluding contour tell us about solid shape? Perception **13** (1984) 321–330
2. Witkin, A.P.: Recovering surface shape and orientation from texture. Artificial intelligence **17**(1-3) (1981) 17–45
3. Malik, J., Rosenholtz, R.: Computing local surface orientation and shape from texture for curved surfaces. IJCV **23**(2) (1997) 149–168
4. Blake, A., Marinos, C.: Shape from texture: estimation, isotropy and moments. Artificial Intelligence **45**(3) (1990) 323–380
5. Fleming, R.W., Torralba, A., Adelson, E.H.: Specular reflections and the perception of shape. Journal of Vision **4**(9) (2004) 798–820
6. Zhang, R., Tsai, P.S., Cryer, J.E., Shah, M.: Shape-from-shading: a survey. IEEE PAMI **21**(8) (1999) 690–706
7. Barron, J.T., Malik, J.: Shape, illumination, and reflectance from shading. IEEE PAMI (2015)
8. Koenderink, J.J., van Doorn, A.J.: Photometric invariants related to solid shape. Optica Acta **27**(7) (1980) 981–996
9. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press, ISBN: 0521540518 (2004)
10. Lowe, D.: Distinctive image features from scale-invariant keypoints. IJCV **60**(2) (2004) 91–110
11. Snavely, N., Seitz, S., Szeliski, R.: Photo tourism: exploring photo collections in 3D. Proc. ACM SIGGRAPH (3) (2006) 835–846
12. Schaffalitzky, F., Zisserman, A.: Multi-view matching for unordered image sets, or "how do i organize my holiday snaps?". In: Proc. ECCV. Volume 1., Springer-Verlag (2002) 414–431
13. Arandjelović, R., Zisserman, A.: Name that sculpture. In: ACM International Conference on Multimedia Retrieval. (2012)
14. Eigen, D., Puhrsch, C., Fergus, R.: Depth map prediction from a single image using a multi-scale deep network. In: NIPS. (2014)
15. Laina, I., Rupprecht, C., Belagiannis, V., Tombari, F., Navab, N.: Deeper depth prediction with fully convolutional residual networks. In: 3D Vision (3DV), 2016 Fourth International Conference on. (2016)
16. Eigen, D., Fergus, R.: Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In: Proc. CVPR. (2015)
17. Chen, W., Fu, Z., Yang, D., Deng, J.: Single-image depth perception in the wild. In: NIPS. (2016)
18. Zoran, D., Isola, P., Krishnan, D., Freeman, W.T.: Learning ordinal relationships for mid-level vision. In: Proc. ICCV. (2015)
19. Vijayanarasimhan, S., Ricco, S., Schmid, C., Sukthankar, R., Fragkiadaki, K.: Sfm-net: Learning of structure and motion from video. arXiv preprint arXiv:1704.07804 (2017)
20. Godard, C., Aodha, O.M., Brostow, G.J.: Unsupervised monocular depth estimation with left-right consistency. In: Proc. CVPR. (2017)
21. Zhou, T., Brown, M., Snavely, N., Lowe, D.G.: Unsupervised learning of depth and ego-motion from video. In: Proc. CVPR. (2017)
22. Ummenhofer, B., Zhou, H., Uhrig, J., Mayer, N., Ilg, E., Dosovitskiy, A., Brox, T.: Demon: Depth and motion network for learning monocular stereo. Proc. CVPR (2017)

23. Choy, C., Xu, D., Gwak, J., Chen, K., Savarese, S.: 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In: Proc. ECCV. (Jul 2016)
24. Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A.: Learning a predictable and generative vector representation for objects. In: Proc. ECCV. (2016)
25. Fan, H., Su, H., Guibas, L.: A point set generation network for 3D object reconstruction from a single image. Proc. CVPR (2017)
26. Sinha, A., Unmesh, A., Huang, Q., Ramani, K.: Surfnet: Generating 3D shape surfaces using deep residual networks. In: Proc. CVPR. (2017)
27. Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J.: Learning a probabilistic latent space of object shapes via 3D generative-adversarial modeling. In: NIPS. (2016) 82–90
28. Soltani, A.A., Huang, H., Wu, J., Kulkarni, T.D., Tenenbaum, J.B.: Synthesizing 3D shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In: Proc. CVPR. (2017)
29. Tulsiani, S., Zhou, T., Efros, A., Malik, J.: Multi-view supervision for single-view reconstruction via differentiable ray consistency. In: Proc. CVPR. (2017)
30. Rezende, D., Eslami, S.M.A., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N.: Unsupervised learning of 3D structure from images. In: NIPS. (2016) 4997–5005
31. Yan, X., Yang, J., Yumer, E., Guo, Y., Lee, H.: Perspective transformer nets: Learning single-view 3D object reconstruction without 3D supervision. In: NIPS. (2016)
32. Gadelha, M., Maji, S., Wang, R.: 3D shape induction from 2D views of multiple objects. arXiv preprint arXiv:1612.05872 (2016)
33. Zhu, R., Kiani, H., Wang, C., Lucey, S.: Rethinking reprojection: Closing the loop for pose-aware shape reconstruction from a single image. Proc. ICCV (2017)
34. Novotny, D., Larlus, D., Vedaldi, A.: Learning 3D object categories by looking around them. Proc. ICCV (2017)
35. Li, Z., Snavely, N.: MegaDepth: Learning single-view depth prediction from internet photos. In: Proc. CVPR. (2018)
36. Hong, J.H., Zach, C., Fitzgibbon, A., Cipolla, R.: Projective bundle adjustment from arbitrary initialization using the variable projection method. In: Proc. ECCV. (2016)
37. Hong, J.H., Zach, C., Fitzgibbon, A.: Revisiting the variable projection method for separable nonlinear least squares problems. In: Proc. CVPR. (2017)
38. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. ACM **24**(6) (1981) 381–395
39. Strang, G.: Linear algebra and its applications. 2 edn. Academic Press, Inc. (1980)
40. Papadopoulo, T., Lourakis, M.I.: Estimating the jacobian of the singular value decomposition: Theory and applications. In: Proc. ECCV. (2000)
41. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch. (2017)
42. Todd, J.T.: The visual perception of 3D shape. Trends in cognitive sciences **8**(3) (2004) 115–121
43. Koenderink, J.J., Van Doorn, A.J., Kappers, A.M.: Surface perception in pictures. Perception & Psychophysics **52**(5) (1992) 487–496
44. Belhumeur, P.N., Kriegman, D.J., Yuille, A.L.: The bas-relief ambiguity. IJCV **35**(1) (1999) 33–44
45. Liwicki, S., Zach, C., Miksik, O., Torr, P.H.: Coarse-to-fine planar regularization for dense monocular depth estimation. In: Proc. ECCV. (2016)
46. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proc. MICCAI. (2015)

47. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. Proc. CVPR (2017)
48. Odena, A., Dumoulin, V., Olah, C.: Deconvolution and checkerboard artifacts. Distill (2016)
49. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: Proc. ICCV. (2017)
50. Arandjelović, R., Zisserman, A.: Smooth object retrieval using a bag of boundaries. In: Proc. ICCV. (2011)
51. Fouhey, D.F., Gupta, A., Zisserman, A.: 3D shape attributes. In: Proc. CVPR. (2016)
52. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
53. Lin, G., Milan, A., Shen, C., Reid, I.: Refinenet: Multi-path refinement networks with identity mappings for high-resolution semantic segmentation. Proc. CVPR (2017)
54. Moulon, P., Monasse, P., Marlet, R., Others: Openmvg. an open multiple view geometry library. https://github.com/openMVG/openMVG
55. Wiles, O., Zisserman, A.: Silnet : Single- and multi-view reconstruction by learning from silhouettes. In: Proc. BMVC. (2017)
56. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An information-rich 3D model repository. Technical Report arXiv:1512.03012 [cs.GR] (2015)
57. Blender Online Community: Blender - a 3D modelling and rendering package. (2017)
58. Choi, S., Zhou, Q.Y., Miller, S., Koltun, V.: A large dataset of object scans. arXiv:1602.02481 (2016)
59. : Pytorch
60. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proc. CVPR. (2016)
61. Silberman, N., Hoiem, D., Kohli, P., Fergus, R.: Indoor segmentation and support inference from RGB-D images. In: Proc. ECCV. (2012)
62. Zhou, Q.Y., Park, J., Koltun, V.: Open3D: A modern library for 3D data processing. arXiv:1801.09847 (2018)

Supplementary Material
3D Surface Reconstruction by Pointillism

Olivia Wiles Andrew Zisserman

Visual Geometry Group,
University of Oxford
{ow,az}@robots.ox.ac.uk

A Overview

Additional qualitative results are given in Section A.1. We elaborate further on the dataset collection in Section B and give sample clusters and images from the Sculpture dataset in Section B.1 and Section B.2. A review of the affine camera model is given in Section C. Finally, we give precise details of the architecture used in Section D.

A.1 Depth predictions on the Sculpture dataset.

Additional qualitative results for 3D shape prediction on the test set of the Sculpture dataset are given in Fig. 1 and on SketchFab in Fig. 2. Qualitative results for LiftNet's depth predictions on ShapeNet are given in Fig. 3.

B Sculpture Dataset: Additional Information

Here we further describe the steps outlined in the paper.

Our dataset is composed of several other datasets [2,3,4,5] and additional images downloaded from the web. These datasets are already divided by artist. We then wish to further subdivide these into a set of works (i.e. sculptures) which are subdivided into a set of clusters which should hold similar viewpoints of the sculpture. The point of the clusters is that we can search for correspondences within clusters as opposed to at the dataset level and there should be many pairs in correspondence within a cluster. For [4,2,5], the sculptures are already divided at the artist level into a set of works. In [3], the dataset is not divided into works, so we use the artist level as the work level (i.e. all images of a given artist are defined to be of one work); this is acceptable, as we divide our dataset at the artist level.

The works are then divided into a set of clusters as follows. [4] is pre-divided and as [2,5] are small there is no need to subdivide their works into clusters (i.e. all images of a given work are defined to be of the same cluster). However, it is necessary to subdivide the works of [3] into clusters, as there may be thousands of images in a work, making correspondence matching slow. This is done as follows. First, a ResNet is trained to classify the works of [4] using the given



Fig. 1: Additional qualitative results for LiftNet (top), COLMAP (middle) and Chen *et al.* [1] (bottom). The input image is shown at the top, then the predicted depth (blue is further away, yellow nearer), and rendered 3D at multiple viewpoints. Zoom in for details.

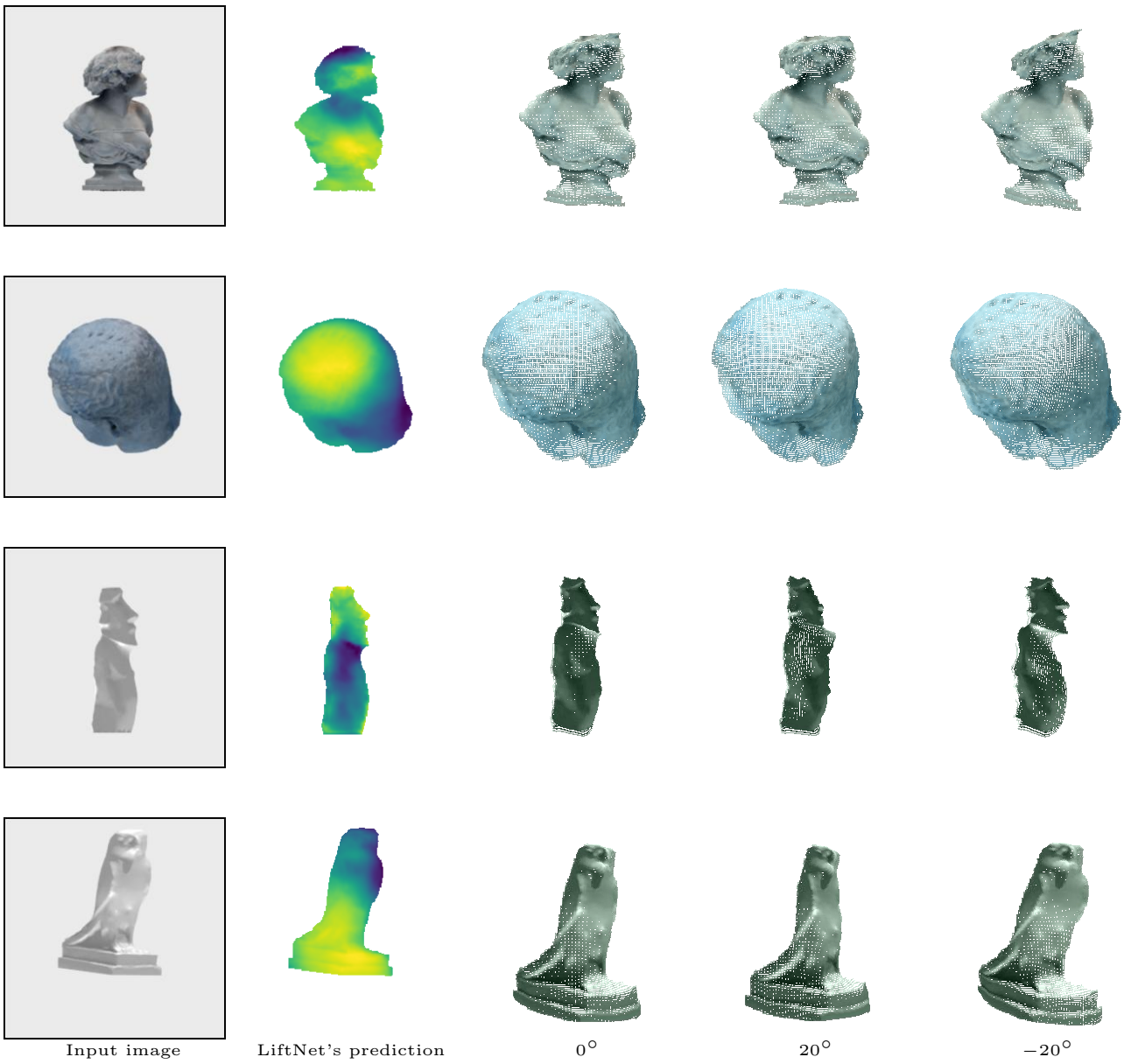


Fig. 2: Additional qualitative results for LiftNet trained on the Sculpture dataset and tested on Sketchfab. For the predicted depth, yellow denotes nearer and blue further away. Additional renderings are also shown.

labels. The fc layer before the final classification is then used to determine the distance between pairs of images using cosine similarity. Each image in a work begins in its own cluster. If the cosine distance between the feature vectors of two images falls below a given threshold then their clusters are merged into one. This is repeated for all pairs.

Additional images from the dataset are given in Section B.1, and some sample clusters are given in Section B.2.

B.1 Images

A selection of images from the dataset is given in figure 4. This illustrates the variety of sculptures in the dataset.

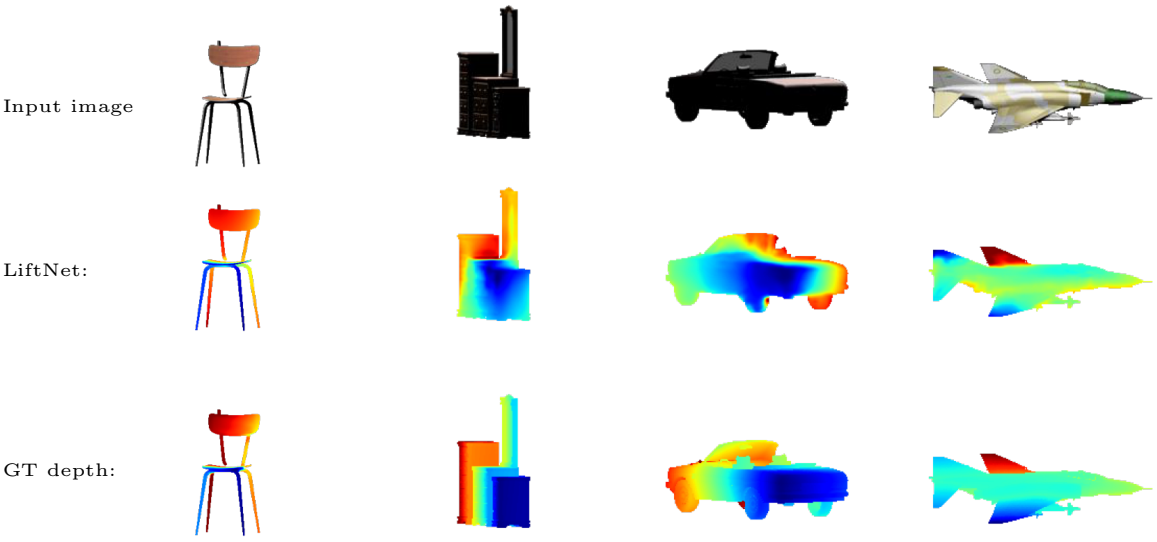


Fig. 3: Additional qualitative results for LiftNet trained and tested on ShapeNet. For the predicted depth, blue denotes nearer and red further away.

B.2 Clusters

Additional clusters are given in figures 5, 6, 7, 8, 9, 10. These illustrate how a given instance of a sculpture may be seen in varying viewpoints and different aspects of the instance constrained by different views. These examples also demonstrate that there may still be outliers in the dataset despite using the affine fundamental matrix to remove invalid correspondences. Our model must be robust to these outliers, necessitating the use of the robustness term.

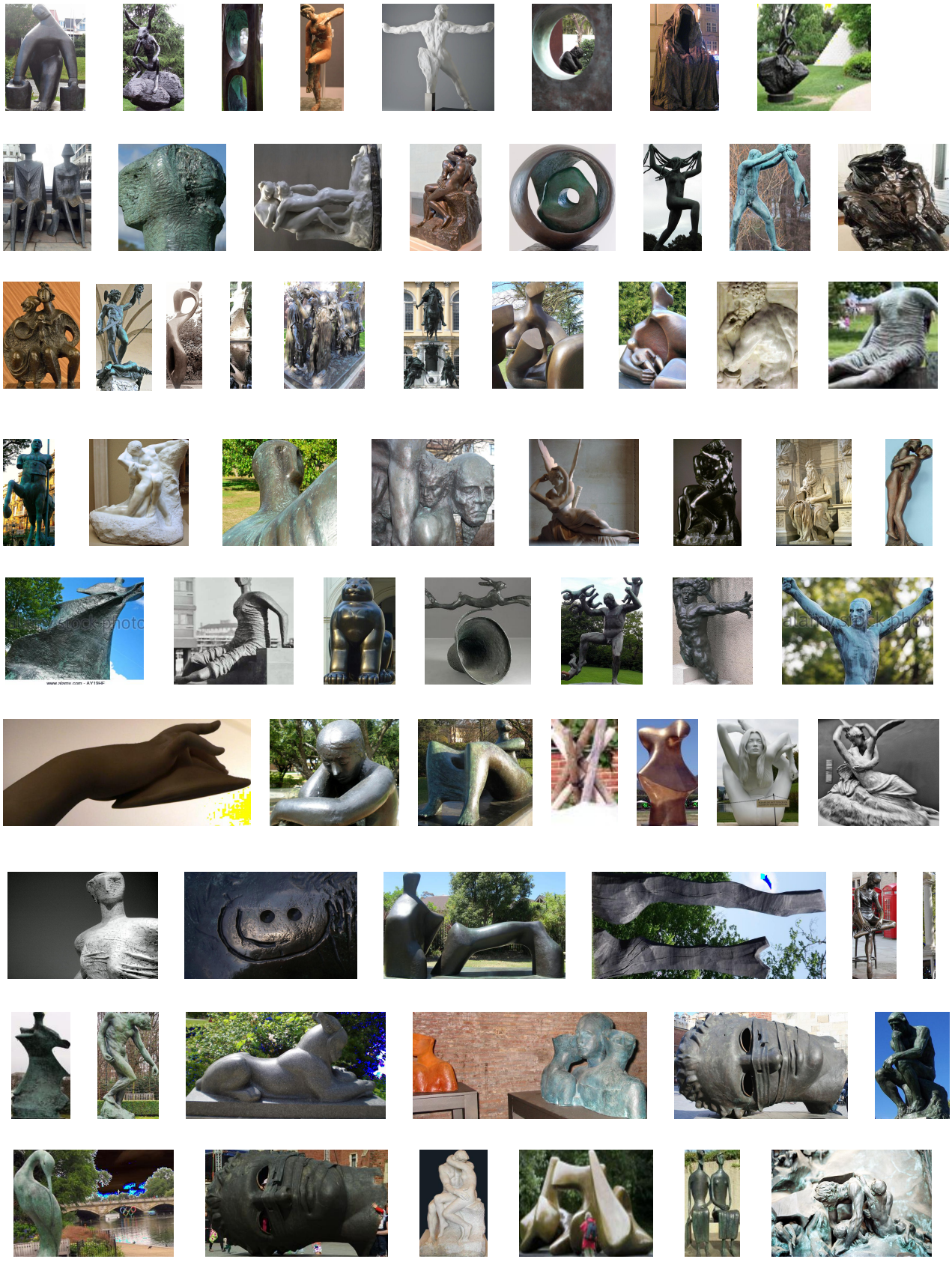


Fig. 4: A selection of images from the Sculpture dataset.

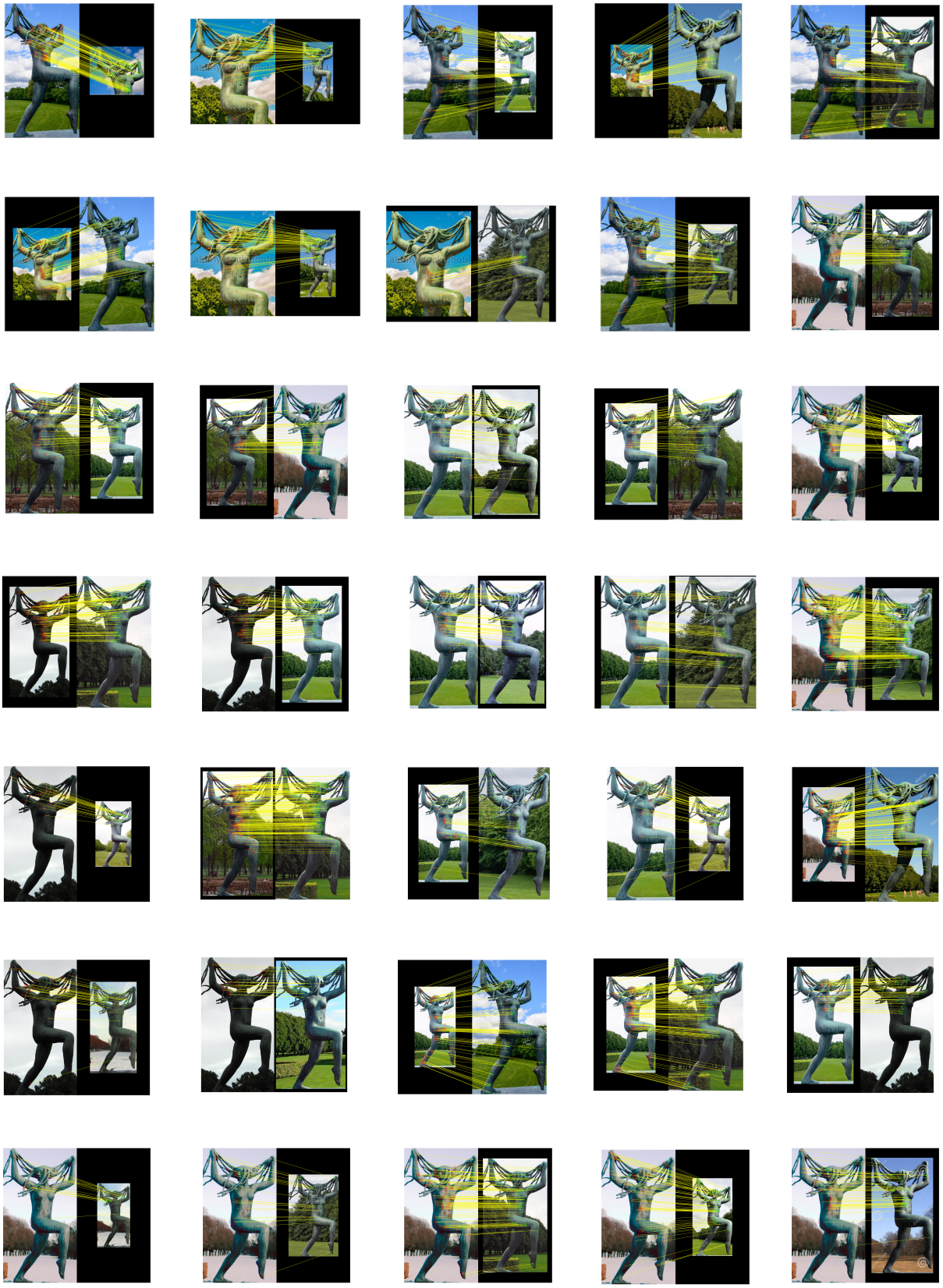


Fig. 5: Gustav Vigeland: *The Dancing Woman*. Shows sample matching pairs from the cluster, in which there are 109 pairs from 85 images.



Fig. 6: Auguste Rodin: *The Crouching Woman*. Shows sample matching pairs from the cluster, in which there are 26 pairs.



Fig. 7: Auguste Rodin: *The Age of Bronze*. Shows sample matching pairs from the cluster, in which there are 41 pairs and 72 images.



Fig. 8: Gustav Vigeland. Shows sample matching pairs from the cluster, in which there are 51 matching pairs and 16 images.

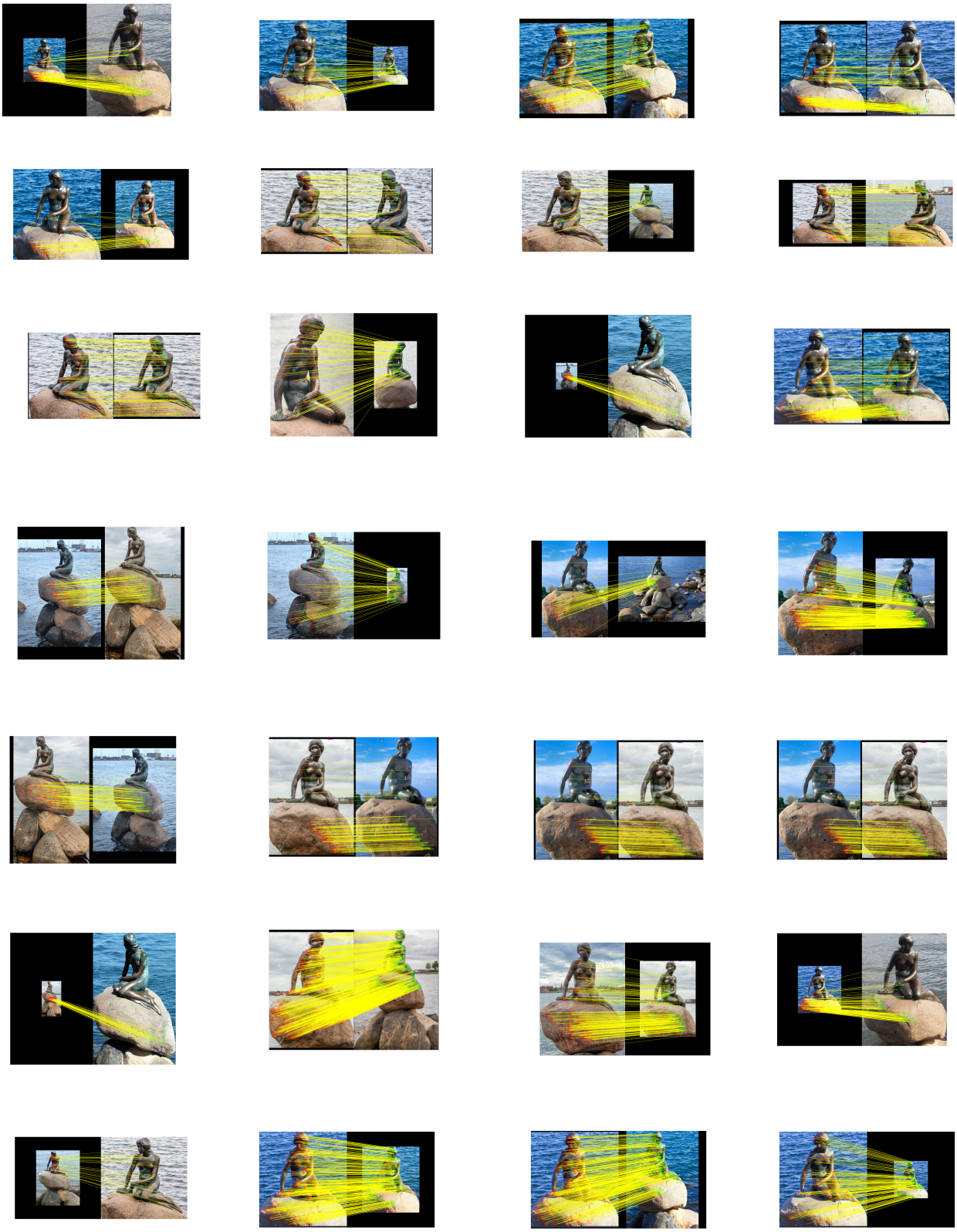


Fig. 9: Edvard Eriksen: *The Little Mermaid*. Shows sample matching pairs from the cluster, in which there are 87 pairs and 97 images.



Fig. 10: Gustav Vigeland: *The Angry Boy*. Shows sample matching pairs from the cluster, in which there are 47 pairs and 81 images.

C *Review of the affine camera model*

In this work, we assume an affine camera model. This model is used for determining the camera when projecting the 3D shape to a new view and in the data preparation phase. The reason for using affine SFM is that it requires fewer correspondences to compute the cameras and 3D structure than in the general (projective) case. Moreover, as has been noted previously [6,7,8,9], the affine case is a very stable and useful approximation to perspective projection.

Here we give a brief overview of the necessary background geometry: we describe our camera model in Section C.1, the resulting affine ambiguity in Section C.2, and how we make use of affine epipolar geometry to remove outliers in Section C.3. This review is based on [10,6,11,7,12,13].

C.1 The factorization algorithm [13]:

We start with this algorithm, as it provides a simple method to obtain a reconstruction of the 3D structure and the (affine) cameras for each view, and defines the notation. Given $n \geq 4$ point correspondences over m views $\mathbf{x}_j^i$, $j = 1, \ldots, n$; $i = 1, \ldots, m$, factorization determines the affine camera matrices $\{\mathtt{M}^i, \mathbf{t}^i\}$ and 3D points $\{\mathbf{X}_j\}$ such that the reprojection error

$$\sum_{ij} \left|\left|\mathbf{x}_j^i - (\mathtt{M}^i \mathbf{X}_j + \mathbf{t}^i)\right|\right|^2$$

is minimized over $\{\mathtt{M}^i, \mathbf{t}^i, \mathbf{X}_j\}$, with $\mathtt{M}^i$ a 2×3 matrix, $\mathbf{X}_j$ a 3-vector, and $\mathbf{x}_j^i = (x_j^i, y_j^i)^\top$ and $\mathbf{t}^i$ are 2-vectors. The algorithm only involves carrying out a SVD decomposition.

The minimal solution is for four point correspondences ($n = 4$), and in this case the reprojection error is zero. The minimal solution is used below to remove outliers using the RANSAC [14] algorithm.

In our case, we have two cameras. We set the first camera matrix to project, orthogonally, into the first view (so $\mathtt{M}^1 = \left(\begin{smallmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{smallmatrix}\right)$ and $\mathbf{t}^1 = [0, 0]^T$). $\mathtt{M}^2, \mathtt{t}^2$ are then determined by solving a system of equations using a simple matrix inverse: $\mathbf{x} = [\mathtt{M}^2; \mathtt{t}^2][\mathbf{X}^T; 1]^T$ (where $[\mathtt{M}^2; \mathtt{t}^2]$ denotes the concatenation of the matrices giving a 2×4 matrix and $[\mathbf{X}^T; 1]$ denotes converting the 3D points into homogeneous coordinates and concatenating them to give a $n \times 4$ matrix).

C.2 The affine ambiguity:

The factorization algorithm [13] demonstrates clearly how the affine ambiguity arises. Given the cameras $\mathtt{M}$ ($\mathtt{M}$ denotes the vertical concatenation of the cameras $\mathtt{M}^1 \ldots \mathtt{M}^m$ to give a $2m \times 3$ matrix) and 3D points $\mathbf{X}$ ($\mathbf{X}$ denotes the horizontal concatenation of the 3D points $\mathbf{X}_1 \ldots \mathbf{X}_n$ giving a $3 \times n$ matrix) determined by the affine factorization algorithm, then for any invertible 3×3 matrix $\mathtt{A}$, $\mathtt{MA}$ and $\mathtt{A}^{-1}\mathbf{X}$ also give a valid decomposition [6]. The $\mathtt{A}$ is the affine ambiguity.

Hence, there is an affine ambiguity in LiftNet's reconstruction. However, in our case $\mathtt{M}^1\mathtt{A}$ must equal $\left(\begin{smallmatrix}1&0&0\\0&1&0\end{smallmatrix}\right)$, so $\mathtt{A}$ is restricted to $\left(\begin{smallmatrix}1&0&0\\0&1&0\\ \alpha&\beta&\gamma\end{smallmatrix}\right)$. This gives the inverse $\mathtt{A}^{-1} = \left(\begin{smallmatrix}1&0&0\\0&1&0\\ -\frac{\alpha}{\gamma}&-\frac{\beta}{\gamma}&\frac{1}{\gamma}\end{smallmatrix}\right)$ This means that the ambiguity in the depth d_i for a 3D point $X_i = [x_i, y_i, d_i]$ is $d_i^* = -\frac{\alpha}{\gamma}x_i - \frac{\beta}{\gamma}y_i + \frac{1}{\gamma}d_i$.

Thus, this ambiguity gives a scaling in depth $\frac{1}{\gamma}$, allowing us to restrict the range of predicted depth values to $[1, -1]$ without restricting the describable surfaces.

C.3 Two view geometry:

Here, we review the epipolar geometry used to remove outliers in the dataset clean-up phase.

The epipolar geometry is represented by the (affine) fundamental matrix. It can be determined by four point correspondences between the two images (or equivalently by using the factorization algorithm, and then computing the fundamental matrix from the camera matrices of the two views). The epipolar geometry is an important workhorse in removing outlier matches using RANSAC: four point correspondences are selected at random and used to compute the epipolar geometry, the remaining correspondences are then tested against this geometry (i.e. do they lie near their epipolar lines), and in this way outlier correspondences (that don't lie on their epipolar lines) can be removed.

D *Architecture used*

LiftNet is based on pix2pix [15], except that the last layer is changed to a tanh layer to so that values are restricted to the range [-1,1] and training does not become unstable. The architecture is illustrated in Fig. 11. This gives a 256×256 depth map for the input image.

References

1. Chen, W., Fu, Z., Yang, D., Deng, J.: Single-image depth perception in the wild. In: NIPS. (2016)
2. Arandjelović, R., Zisserman, A.: Smooth object retrieval using a bag of boundaries. In: Proc. ICCV. (2011)
3. Arandjelović, R., Zisserman, A.: Name that sculpture. In: ACM International Conference on Multimedia Retrieval. (2012)
4. Fouhey, D.F., Gupta, A., Zisserman, A.: 3D shape attributes. In: Proc. CVPR. (2016)
5. Knapitsch, A., Park, J., Zhou, Q.Y., Koltun, V.: Tanks and temples: Benchmarking large-scale scene reconstruction. ACM Transactions on Graphics **36**(4) (2017)
6. Shapiro, L.S.: Affine Analysis of Image Sequences. PhD thesis, University of Oxford, England (1993)

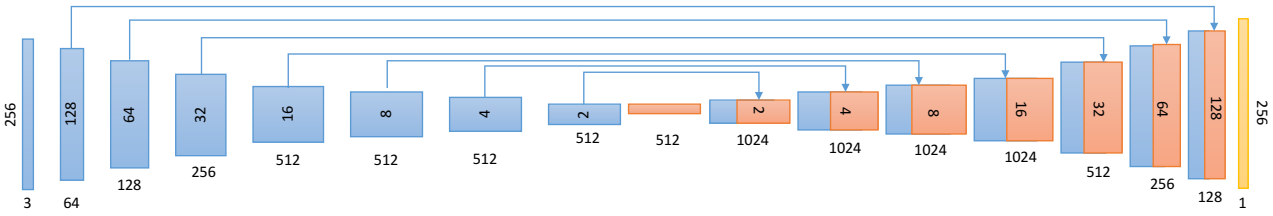


Fig. 11: The LiftNet architecture based on pix2pix. The channel sizes are given below each block and the size of the filters within (they are all square). In the encoder (blue), each convolution is followed by a leaky ReLU with factor 0.2 and then a batch-norm layer (except for the first which has no batch-norm). The convolutional filter sizes are 4×4 and the stride/padding is 2/1. In the decoder (red), the layers from the encoder are concatenated using skip connections with the corresponding decoder layer, doubling the channel size in the decoder. In all decoder units the sequence of layers is: relu, bilinear upsample, batch-norm, concatenation (with the corresponding encoder layer). The last layer (yellow) does not have a concatenation. The convolutional filter sizes are 3×3 and the stride/padding is 1/1. The final 256×256 result is passed through a tanh layer to give LiftNet's depth prediction.

7. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision. Second edn. Cambridge University Press, ISBN: 0521540518 (2004)
8. Hong, J.H., Zach, C., Fitzgibbon, A., Cipolla, R.: Projective bundle adjustment from arbitrary initialization using the variable projection method. In: Proc. ECCV. (2016)
9. Hong, J.H., Zach, C., Fitzgibbon, A.: Revisiting the variable projection method for separable nonlinear least squares problems. In: Proc. CVPR. (2017)
10. Koenderink, J.J., van Doorn, A.J.: Affine structure from motion. J. Opt. Soc. Am. A **8**(2) (1991) 377–385
11. Mundy, J., Zisserman, A.: Geometric Invariance in Computer Vision. MIT Press (1992)
12. Shapiro, L., Zisserman, A., Brady, J.M.: Motion from point matches using affine epipolar geometry. In: Proc. ECCV. LNCS 800/801, Springer-Verlag (1994)
13. Tomasi, C., Kanade, T.: Shape and motion from image streams under orthography: A factorization approach. IJCV **9**(2) (Nov 1992) 137–154
14. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Comm. ACM **24**(6) (1981) 381–395
15. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. Proc. CVPR (2017)

Chapter 6

Self-Supervised Learning of a Facial Attribute Embedding from Video

This work was presented as an oral and published at BMVC 2018.

This work introduces a self-supervised method for learning an embedding that encodes pose, expression, and 3D information specific to the class of faces.

Contributions of Olivia Wiles

1. Worked on writing the paper
2. Developed ideas for the self-supervised model and improvements: curriculum, multi-view strategy
3. Ran experiments for the model and the improvements
4. Developed ideas and tasks for using the model in a downstream manner
5. Developed testing frameworks for the model: especially landmark estimation and nearest neighbour visualisation
6. Evaluated the model on the downstream tasks

Self-supervised learning of a facial attribute embedding from video

Olivia Wiles*
ow@robots.ox.ac.uk
A. Sophia Koepke*
koepke@robots.ox.ac.uk
Andrew Zisserman
az@robots.ox.ac.uk

Visual Geometry Group
University of Oxford
Oxford, UK

Abstract

We propose a self-supervised framework for learning facial attributes by simply watching videos of a human face speaking, laughing, and moving over time. To perform this task, we introduce a network, **F**acial **A**ttri**b**utes-**Net** (FAb-Net), that is trained to embed multiple frames from the same video face-track into a common low-dimensional space. With this approach, we make three contributions: first, we show that the network can leverage information from multiple source frames by predicting confidence/attention masks for each frame; second, we demonstrate that using a curriculum learning regime improves the learned embedding; finally, we demonstrate that the network learns a meaningful face embedding that encodes information about head pose, facial landmarks and facial expression – i.e. facial attributes – *without* having been supervised with any labelled data. We are comparable or superior to state-of-the-art self-supervised methods on these tasks and approach the performance of supervised methods.

1 Introduction

Babies and children are highly perceptive to the facial expressions of the people they interact with [14, 18]. The ability to understand and respond to changes in people's emotional state is similarly important for computer vision systems and affective systems when interacting with a human user. Thus being able to predict head pose and expression is of vital importance.

Recently, leveraging deep learning has led to state-of-the-art results on a variety of tasks such as emotion recognition and facial landmarks detection. Despite these advances, supervised methods require large amounts of labelled data which may be expensive or difficult to obtain in realistic, unconstrained settings, or necessitate assigning data to ill-defined categories. For example, categorising emotions with three human annotators leads to only 46% agreement [2], and labelling pose in the wild is notoriously difficult. Moreover, performing each task independently does not leverage the fact that detecting landmarks requires understanding pose and facial features, which in turn correspond to expression.

Consequently, we consider the following question: is it possible to learn an embedding of facial attributes that encodes landmarks, pose, emotion, etc. in a self-supervised manner without *any* hand labelling? The learned embedding can then be used for another task

© 2018. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

* Equal contribution.

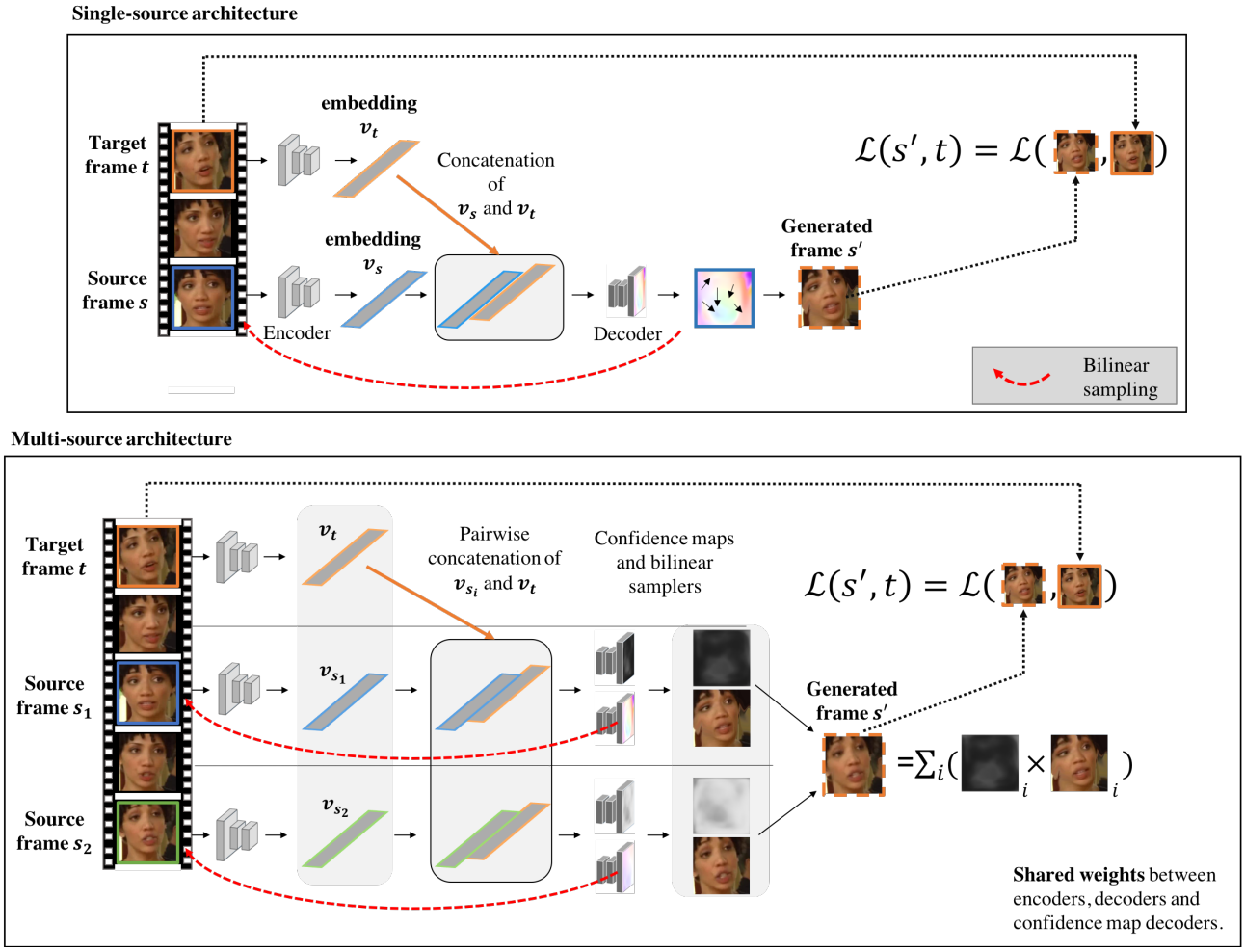


Figure 1: An overview of FAb-Net. In the single-source case (top), the encoder-decoder architecture takes one source frame and a target frame as inputs and learns to generate the target frame. The 256-dimensional outputs of the encoders – the source and target attribute embeddings – are concatenated and input to the decoder. The decoder predicts the point-to-point correspondence from the source frame to the target frame, and RGB values for the generated frame are then obtained from the source frame using a bilinear sampler. The network is trained with an $L1$ loss between the generated frame and the target frame. In the multi-source case (bottom), the decoder also predicts a confidence map, and the confidence maps are used to weight the contributions of the different source frames.

(e.g. landmark, pose, and expression prediction) using a linear layer. To do this, we contribute FAb-Net, a self-supervised framework for learning a low-dimensional face embedding for facial attributes (Section 3). We take advantage of video data which contains a large collection of images of the same person from different viewpoints and with varied expressions. Given only the embeddings corresponding to a source and target frame, the network is tasked to map the source frame to the target frame by predicting a flow field between them. This proxy task forces the network to distil the information required to compute the flow field (e.g. the head pose and expression) into the source and target embeddings. After explaining the setup for a single source frame in Section 3.1, we introduce our additional contributions: a method for leveraging multiple frames in order to improve the learned embedding in Section 3.2; and how a curriculum strategy for training FAb-Net in Section 4 can be used to improve performance.

The learned embedding is extracted and used for a variety of tasks such as landmark

detection, pose regression, and expression classification in Section 5 by simply learning a linear layer. Our results on these tasks are comparable or superior to other self-supervised methods, approaching the performance of supervised methods. These experiments verify the hypothesis that our self-supervised framework learns to encode facial attributes which are useful for a variety of tasks. Finally, the method is tested qualitatively by using the learned embedding to retrieve images with similar facial attributes across different identities.

2 Related Work

Self-supervised learning. Self-supervised methods such as [12, 38, 43, 62] require no manual labelling of the images; instead, they directly use image data to provide proxy supervision for learning good feature representations. The benefit of these approaches is that the features learned using large quantities of available data can be transferred to other tasks/domains which have less or even no annotated data. To provide further supervision from image data, the images themselves can be transformed via a synthetic warp or rotation and the network trained to recognise the rotation [19] or to learn equivariant pixel embeddings [40, 51, 52].

Of more direct relevance to our training framework are self-supervised frameworks that use video data ([1, 8, 11, 16, 17, 21, 22, 30, 34, 44, 55, 56, 58, 61]). Our approach builds in particular on those that use frame synthesis [8, 11, 22, 44, 56, 58], though for us synthesis is a proxy task rather than the end goal. Note, unlike [16, 30, 34, 55], we do not make use of the temporal ordering information inherent in a video; nor do we predict future frames conditioned on a number of past frames [44], or explicitly predict the motion between frames as a convolutional kernel [22, 58], or condition the generation on another modality (e.g. voice [8]). Instead, we treat the frames as an unordered set, and propose a simple formulation: that by embedding the source and target frames in a common space and conditioning the transformation from source to target frame on these embeddings, the learned embeddings must learn to encode the relevant modes of variation necessary for the transformation.

Concurrent to our work, Zhang *et al*. [64] and Jakab *et al*. [21] build on [51] by using the discovered landmarks to reconstruct the original image. However, unlike these works, we do not place any constraints on the learned representation – such as an explicit representation that encodes landmarks as heatmaps.

Supervised learning of face embeddings. Given *known* (labelled) attribute information, e.g. for pose or expression, the embedding can be learned by training in a supervised manner to directly predict the attribute [28, 32, 46], or to generate images (of faces, cars, or other classes) at a new, *known* pose, expression, etc. [13, 25, 53, 59, 68]. Another way of supervising a face embedding is to explicitly learn the parameters of a 3D morphable model (3DMM) [7]. As fitting a 3DMM is relatively expensive, [3, 50] learn this end-to-end using either landmarks or a photometric error as supervision. However, unlike our method, these methods require either ground truth labels or a morphable model which fixes the modes of variation and the embedding.

An interesting half-way point is weak supervision, where the learned object or face embedding is conditioned for instance on object labels [39] or weather/geo-location information [31] respectively. This requires additional meta-data, but results in embeddings that can represent attributes such as age and expression for faces or keypoints for objects.

3 Method

The aim is to train a network to learn an embedding that encodes facial attributes in a self-supervised manner, without any labels. To do this, the network is trained to generate a target frame from one or multiple source frames by learning how to transform the source into the target frame. The source and target frames are taken from the same face-track of a person speaking, i.e. the frames are of the same identity but with different expressions/poses. An overview of the architecture is given in Fig. 1 and further described for a single source frame in Section 3.1 and for multiple source frames in Section 3.2 (additional details are given in the supp. material).

3.1 Single-source frame architecture

The input to the network is a source frame s and a target frame t from the same face-track. These are passed through encoders with shared weights which learn a mapping f from the input frames to a 256-dimensional vector embedding (as shown in Fig. 1). The embeddings corresponding to the target and source frames are $v_t = f(t)$ and $v_s = f(s)$ respectively. The source and target embeddings are concatenated to give a 512-dimensional vector which is upsampled via a decoder. The decoder learns a mapping g from the concatenated embeddings to a bilinear grid sampler, which samples from the source frame to create a new, generated frame $s' = g(v_t, v_s)(s)$. Precisely, g predicts offsets $(\delta x, \delta y)$ for each pixel location (x, y) in the target frame; the generated frame s' at location (x, y) is obtained by sampling from the source frame s according to these offsets : $s'(x, y) = s(x + \delta x, y + \delta y)$. The network is trained to minimise the $L1$ loss between the generated and the target frame: $\mathcal{L}(s', t) = ||t - s'||_1$.

This setup enforces that the embeddings v_s and v_t represent facial attributes of the source and target frames respectively since the decoder maps from the source frame s to generate the frame s' (i.e. it uses pixel RGB values from the source frame s to create the generated s' – a similar formulation has been proposed concurrently to this work by [54]). As the decoder is a function of the target and source attribute embeddings, and the decoder is the only place in the network where information is shared, the target attribute embedding must encode information about expression and pose in order for the decoder to know where to sample from in the source frame and where to place this information in the generated frame.

3.2 Multi-source frames architecture

While using two frames for training enforces that the network learns a high-quality embedding, additional source frames can be leveraged to improve the learned embedding. This is achieved by also predicting a confidence heatmap – a 1 channel image – for each source frame via an additional decoder. The heatmaps denote how confident the network is of the flow at each pixel location – e.g. if the source frame has a very different pose than the target frame, the confidence heatmap would have low certainty. Moreover, it can express this for sub-parts of the image; if the mouth is closed in the source but open in the target frame, the confidence heatmap can express uncertainty in this region. The confidence heatmaps C_i are combined pixel-wise for each source frame s_i using a soft-max operation. For n source frames, the loss function to be minimised is given as $\mathcal{L} = ||t - \frac{\sum_{i=1}^{n} e^{C_i} * (g(v_t, v_{s_i})(s_i))}{\sum_{i=1}^{n} e^{C_i}}||_1$.

4 Curriculum Strategy

The training of the network is divided into stages, so that knowledge can be built up over time as the examples given become progressively more difficult, as inspired by [5, 27]. The loss computed by a forward pass is used to rank samples (i.e. source and target frame pairs) in the batch according to their difficulty in a manner similar to [33, 37, 48, 49]. However, these methods use only the most difficult samples, which was found to stop our network from learning. Similarly to [37], using progressively more difficult samples proved crucial for the strategy's success.

Given a batch size of N randomly chosen samples, i.e. source and target frame pairs, a forward pass is executed and the loss for each sample computed. The samples are ranked and sorted according to this loss. Initially the loss is back-propagated only on the samples in the batch which are in the 50th percentile (i.e. the $0.5N$ samples with the lowest loss computed by the forward pass). These are assumed to be easier samples. When the loss on the validation set plateaus, the subset to be back-propagated on is shifted by 10 (e.g. the samples in the 10th-60th percentile range). This is repeated 4 times until the samples being back-propagated on fall into the 40th-90th percentile range. At this point the curriculum strategy is terminated, as it is assumed that the samples in the 90-100th range are too challenging or may be problematic (e.g. there is a large shift in the background which is too challenging to learn).

5 Experiments

In this section, we evaluate the network and the learned embedding. In Section 5.1, the performance of using FAb-Net's learned representation is compared to that of state-of-the-art self-supervised and supervised methods on a variety of tasks: facial landmark prediction, head pose regression and expression classification. Section 5.2 discusses the benefit of using additional source frames, and Section 5.3 shows how the learned representation can be used for retrieving images with similar facial attributes.

Training. The model is trained on the VoxCeleb1 and VoxCeleb2 video datasets [9, 36]; we refer to the combined datasets as VoxCeleb+. The VoxCeleb+ dataset consists of videos of interviews containing more than 1 million utterances of around 7,000 speakers. The frames are extracted at 1 fps. The frames are cropped, resized to 256×256, and the identities are randomly split into train/val/test (with a split of 75/15/10).

The models are trained in PyTorch [42] using SGD, an initial learning rate of 0.001, and momentum 0.9. When using the curriculum strategy described in Section 4, the batchsize is $N = 32$, else $N = 8$. The learning rate is divided by a factor of 10 when the loss on the validation set plateaus. (If the curriculum strategy is used, the learning rate is updated only when the 40-90th percentile is considered.) This is repeated until the loss converges. Further details about the training can be found in the supp. material.

5.1 Using the embedding for regression and classification

First, we investigate the representation learned in our embedding and evaluate whether it indeed encodes facial attributes by challenging it to predict three different attributes: landmarks, pose, and expression.

Setup. Given a network trained on VoxCeleb+, a linear regressor or classifier is trained from the learned embedding to the output task. The linear regressor/classifier consists of two layers: batch-norm [20] followed by a linear fully connected layer with no bias. The regression tasks are trained using a MSE loss. The classification tasks are trained with a cross-entropy loss. The parameters of the encoder are fixed while the two additional layers are trained on the training set of the target dataset using Adam [23], a learning rate of 0.001, $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

5.1.1 Baselines

Self-supervised. There are prior publications on using self-supervision for landmark prediction on the datasets we evaluate on, but none for predicting emotion on standard datasets. Consequently, we implement an autoencoder and a set of state-of-the-art self-supervised methods [19, 63] for object detection and segmentation. The baselines are trained using the same architecture as FAb-Net but with their associated loss functions and training objectives. For [63], the regression loss for both the L and ab channels is used. These models are trained on VoxCeleb+ until convergence, with the same training parameters and data augmentation as FAb-Net. More details are given in the supp. material.

VGG-Face descriptor. We additionally compare to the *VGG-Face descriptor* which is obtained from the 4096-dimensional FC7 features from a VGG-16 network trained on the VGG-Face dataset [41]. Contrary to popular belief, it has been recently shown that a network trained for identity does retain information about other facial attributes [10, 15]. We use the VGG-Face descriptor to learn a linear regression/classification layer to the desired attribute task. This provides a strong baseline, and the results obtained confirm the finding that a network trained for identity does indeed encode expression and to some extent also pose information. However, note that unlike our method, this face descriptor requires a large dataset of *labelled* face images for training.

5.1.2 Results

Facial landmarks. Facial landmark locations are regressed from the learned embedding and compared to state-of-the-art methods on MAFL [66] and the more challenging 300-W [47] datasets. The evaluation is performed as outlined in [51, 66], and the errors given in inter-ocular distance. For MAFL, 5 facial landmarks are regressed for 19k/1k train/test images. For 300-W, 68 landmarks are regressed for 3148/689 train/test images which are obtained (as described in [51]) from combining multiple datasets [4, 67, 70].

The results are reported in Table 1 and some qualitative results are visualised in Fig. 2 and Fig. 3. These results first demonstrate that fine-tuning with additional views and our curriculum strategy improve the embedding learned by FAb-Net. Second, these results show that our method performs competitively or better than state-of-the-art unsupervised landmark detection methods, better than the VGG-Face descriptor baseline and competitively with state-of-the-art supervised methods. This is achieved even though the other self-supervised methods [21, 51, 52, 64] are explicitly engineered to detect landmarks whereas our method is not. In addition to that, our method is able to bridge the domain gap between VoxCeleb+ and CelebA [32] (the other self-supervised methods that we compare to are pre-trained on CelebA).

Pose. The learned embedding is used for pose prediction and compared to a supervised method [26] and to using the VGG-Face descriptor. To perform the evaluation, the linear

Method	300-W	MAFL
Self-supervised		
Trained on VoxCeleb+		
FAb-Net	6.31	3.78
FAb-Net w/ curric.	5.73	3.49
FAb-Net w/ curric., 3 source frames	**5.71**	3.44
Trained on CelebA		
Jakab *et al.* [21] (2018)	–	**3.08**
Zhang *et al.* [64] (2018)	–	3.15
Thewlis *et al.* [51] (2017)	9.30	6.67
Thewlis *et al.* [52] (2017)	7.97	5.83
Supervised		
Trained on CelebA		
MTCNN [65] (2014)	–	**5.39**
LBF [45] (2014)	6.32	–
CFSS [69] (2015)	5.76	–
cGPRT [29] (2015)	5.71	–
DDN [60] (2016)	5.65	–
TCDCN [66] (2016)	5.54	–
RAR [57] (2016)	**4.94**	–
VGG-Face descriptor [41]	11.16	5.92

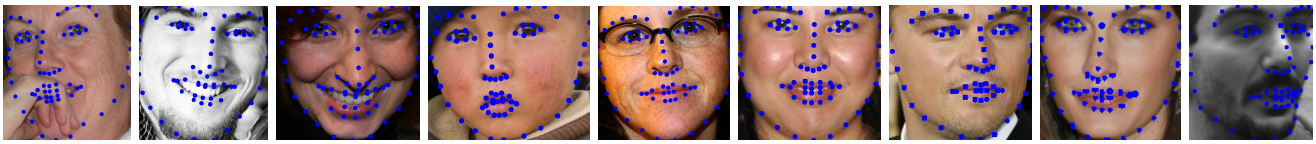
Table 1: Landmark prediction error on 300-W and MAFL datasets. Lower is better.

Method	Roll	Pitch	Yaw	MAE
Self-supervised				
FAb-Net	5.54°	7.84°	12.93°	8.77°
FAb-Net w/ curric.	5.33°	7.21°	11.34°	7.96°
FAb-Net w/ curric., 3 source frames	**5.14°**	**7.13°**	**10.70°**	**7.65°**
Supervised				
VGG-Face descriptor [41]	**8.24°**	8.36°	18.35°	11.65°
KEPLER [26] (2017)	8.75°	**5.85°**	**6.45°**	**7.02°**

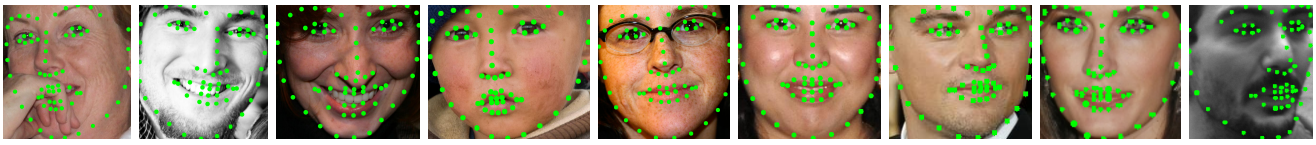
Table 2: Pose prediction error on the AFLW test set from [26]. Lower is better.



Figure 2: Landmark prediction visualisation for FAb-Net on the MAFL dataset. A dot denotes ground truth and the cross FAb-Net's prediction. A failure case is shown to the right.



(a) Ground truth landmarks



(b) FAb-Net's predicted landmarks

Figure 3: Landmark prediction visualisation for FAb-Net on the 300-W dataset.

regression is trained from the given embedding to head pose labels using the AFLW dataset [24], but after leaving out the 1,000 images of the AFLW test set from [26]. As can be seen in Table 2, FAb-Net performs better in predicting the roll angle, and the MAE is comparable to [26] which is supervised with head pose labels. Furthermore, our embedding outperforms the VGG-Face descriptor which is trained on identities; i.e. our learned embedding encodes more information about head pose.

Expression. We evaluate the performance of our learned embedding for expression estimation on two datasets: AffectNet [35] and EmotioNet [6], which both contain over 900,000 images. These datasets are taken 'in-the-wild' as opposed to in a constrained environment. AffectNet contains 8 facial expressions (neutral, happy, sad, surprise, fear, disgust, anger, contempt) and EmotioNet contains 11 action units (AUs) (combinations of AUs correspond

	AUC for different AUs											
	1	2	4	5	6	9	12	17	20	25	26	avg.
Self-supervised												
FAb-Net	72.0	68.9	73.2	69.4	88.2	78.6	89.5	71.0	75.9	81.4	72.0	76.4
FAb-Net w/ curriculum	73.4	71.8	75.3	67.8	90.4	78.8	91.9	72.4	74.5	**83.7**	73.3	77.6
FAb-Net w/ curriculum, 3 source frames	**74.1**	**72.3**	**75.8**	**68.8**	**90.7**	**81.8**	**92.5**	**73.7**	**77.2**	83.6	**73.6**	**78.6**
Gidaris *et al*. [19]	68.6	64.0	72.8	70.0	83.9	78.1	83.8	68.4	72.6	73.1	67.2	72.9
SplitBrain [63]	65.5	59.8	66.7	60.8	71.8	65.8	73.3	64.5	57.4	68.1	61.1	65.0
Autoencoder	67.2	60.5	70.1	65.1	79.6	70.4	80.1	68.3	66.5	70.5	64.1	69.3
Supervised												
VGG-Face descriptor [41]	**81.8**	**83.0**	83.5	81.8	92.0	**90.9**	95.7	**80.6**	85.2	86.5	73.0	84.9
VGG-11 (from scratch)	74.7	77.2	**85.8**	**83.7**	**93.8**	89.7	**97.5**	78.3	**86.9**	**96.4**	**81.5**	**86.0**

Table 3: Expression classification results for state-of-the-art self-supervised and supervised methods on EmotioNet [6] for multiple facial action units (AUs). Higher is better for AUC.

	AUC								
	Neutral	Happy	Sad	Surprise	Fear	Disgust	Anger	Contempt	avg.
Self-supervised									
FAb-Net	70.0	87.6	68.8	75.5	76.5	70.0	73.2	71.2	74.2
FAb-Net w/ curric.	71.5	90.0	70.8	78.2	77.4	72.2	75.7	72.1	76.0
FAb-Net w/ curric., 3 source frames	**72.3**	**90.4**	**70.9**	**78.6**	**77.8**	**72.5**	**76.4**	**72.2**	**76.4**
Gidaris *et al*. [19]	67.8	84.9	69.0	73.9	75.7	69.8	71.5	68.7	72.7
SplitBrain [63]	63.9	74.7	64.2	61.3	68.3	58.6	68.2	62.8	65.3
Autoencoder	65.8	80.0	64.7	66.1	70.6	63.4	68.3	65.0	68.0
Supervised									
VGG-Face descriptor [41]	75.9	92.2	80.5	81.4	82.3	81.4	81.2	77.1	81.5
AlexNet [35]	–	–	–	–	–	–	–	–	**82**

Table 4: Expression classification results for state-of-the-art self-supervised and supervised methods on AffectNet [35]. Higher is better for AUC.

to facial expressions).

Both of these datasets were organised for challenges with a held out, unreleased test set. Therefore, the train set is subdivided into two subsets; one is used for training and the other for validation. The validation set of the original dataset is used to test the different models. The linear classifier for EmotioNet is trained with a binary cross-entropy loss for each AU, whereas for AffectNet, a cross-entropy loss is used. Both training datasets are highly imbalanced. As a result, the examples from the under-represented classes are re-weighted inversely proportionally to the class frequencies to penalise the loss more heavily for mis-classifying images of the under-represented classes.

The embedding learned by FAb-Net is compared to a number of self-supervised and supervised methods by measuring the Area Under the ROC curve (AUC). For each class (e.g. emotion or AU), the AUC is computed independently and the result is averaged over all classes. The results are reported in Table 3 and Table 4 for EmotioNet and AffectNet respectively showing that our network performs better than other self-supervised methods over both metrics when given the same training data. This is supposedly due to the fact that the network must learn to transform the source frame in order to generate the target frame. As parts of the face move together (e.g. an eyebrow raise or the lips when the mouth opens), the embedding must learn to encode information about facial features and thereby encode expression. Interestingly, the autoencoder performs well, presumably due to the restricted nature of this domain.

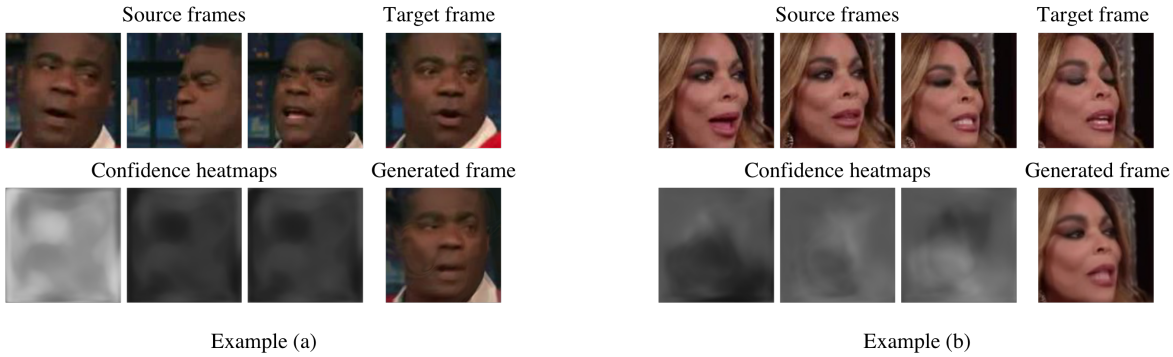


Figure 4: Confidence heatmaps learned by FAb-Net. Higher intensity corresponds to higher confidence. The network chooses the frames with most similar poses to draw from and ignores those with less similar poses (see Example (a)). In Example (b), the mouth has higher confidence in the third source frame allowing the network to re-construct the teeth that are present in the target frame. More examples can be found in the supp. material.

FAb-Net is also not far off supervised methods despite the domain shift; VoxCeleb+ consists only of people being interviewed (and consequently with mostly neutral/smiling faces), so it does not include the range/extremity of expressions found in AffectNet or EmotioNet. Finally, it can be observed that the VGG-Face descriptor trained to predict identities does surprisingly well at predicting emotion.

Discussion. FAb-Net has achieved impressive performance, as most self-supervised methods when transferred to another task have a large gap in comparison to supervised methods. There is no gap or a small gap for the smaller datasets (landmarks/pose) and the model approaches supervised performance for the larger datasets (expression).

5.2 What is the benefit of additional source frames?

The previous sections have shown that using additional source frames improves performance. This is at the expense of performing additional forward passes through the encoder (in this case two). Given enough GPU memory, these forward passes can be done in parallel, affecting only the memory requirements and not the computational speed.

Using multiple source frames is further investigated by visualising confidence heatmaps for a given set of source frames in Fig. 4. The confidence heatmaps allow images with more similar pose to be used for creating the generated frame. Furthermore, the network can focus on one frame for generating a part of the face (e.g. the mouth) and on another one for a different part.

5.3 Image retrieval

This section considers an application of the learned embedding: retrieving images with similar facial attributes (e.g. pose) but across different identities. To perform this task, a subset of 10,000 randomly sampled test images from VoxCeleb+ is obtained. For a given query image, all other images (the gallery) are ranked based on their similarity to the query image using the cosine similarity metric between the corresponding embeddings. For a given query image Q, the embedding x_q is extracted by performing a forward pass through the network. Similarly, the embedding x_i is extracted for each image I_i in the gallery. Each image I_i is then ranked according to the cosine similarity between x_q and x_i. If the network does indeed

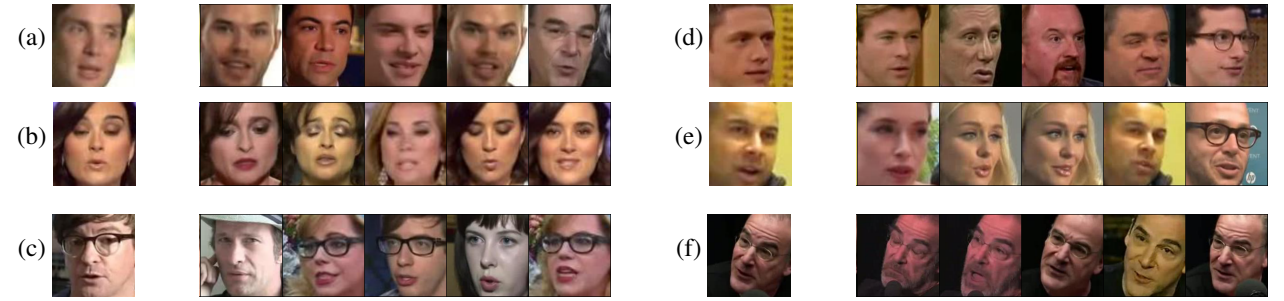


Figure 5: Retrieval results using the embedding learned by FAb-Net. The embedding captures similar visual attributes since gallery images with similar facial attributes are retrieved for a given query image. The retrieved images have similar pose to the query in all cases, and the expression similarity can be seen for example in (b) with the eyes shut, and (a) with the mouth slightly open. Please refer to the supp. material for additional examples.

encode salient information about facial attributes, the cosine similarity can be used to identify images with similar poses and facial attributes. For a set of query images, the results are visualised in Fig. 5. From these results it is again affirmed that our embedding encodes information about facial attributes, as the retrieved images have poses and expressions similar to those of the query images. Note, the embedding is largely unaffected by facial decorations (e.g. glasses) and identity, as these do not change within a face-track and so do not need to be learned in order to predict the transformation.

6 Conclusion

We have introduced FAb-Net: a self-supervised framework for learning facial attributes from videos. Our method learns about pose and expression by watching faces move and change over a large number of videos without *any* hand labels. The features of our trained network can then be used to predict pose, landmarks, and expression on other datasets (despite the domain shift) by just training a linear layer on top of the learned embedding. The features have been shown to be comparable or superior performance to self-supervised and supervised methods on a variety of tasks. This is impressive as generally the performance of self-supervised methods has been found to be worse than that of supervised methods, yet our method is indeed competitive/superior to supervised methods for pose regression and facial landmark detection, and approaches supervised performance on expression classification.

7 Acknowledgements

The authors would like to thank James Thewlis for helpfully sharing code and datasets. This work was funded by an EPSRC studentship and EPSRC Programme Grant Seebibyte EP/M013774/1.

References

[1] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proc. ICCV*, 2015.

[2] E. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang. Training deep networks for facial expression recognition with crowd-sourced label distribution. In *Proceedings of the 18th ACM International Conference on Multimodal Interaction*, pages 279–283. ACM, 2016.

[3] A. Bas, P. Huber, W. A. P. Smith, M. Awais, and J. Kittler. 3D morphable models as spatial transformer networks. In *Proc. ICCV Workshop on Geometry Meets Deep Learning*, 2017.

[4] P. N. Belhumeur, D. W. Jacobs, D. J. Kriegman, and N. Kumar. Localizing parts of faces using a consensus of exemplars. *IEEE PAMI*, 2013.

[5] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proc. ICML*, 2009.

[6] C. F. Benitez-Quiroz, R. Srinivasan, and A. M. Martinez. EmotioNet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild. In *Proc. CVPR*, 2016.

[7] V. Blanz, S. Romdhani, and T. Vetter. Face identification across different poses and illumination with a 3D morphable model. In *Proc. AFGR*, 2002.

[8] J. S. Chung, A. Jamaludin, and A. Zisserman. You said that? In *Proc. BMVC.*, 2017.

[9] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

[10] F. Cole, D. Belanger, D. Krishnan, A. Sarna, I. Mosseri, and W. T. Freeman. Synthesizing normalized faces from facial identity features. In *Proc. CVPR*, 2017.

[11] E. L. Denton and V. Birodkar. Unsupervised learning of disentangled representations from video. In *NIPS*, 2017.

[12] C. Doersch, A. Gupta, and A. A. Efros. Unsupervised visual representation learning by context prediction. In *Proc. ICCV*, 2015.

[13] A. Dosovitskiy, J. T. Springenberg, and T. Brox. Learning to generate chairs with convolutional neural networks. In *Proc. CVPR*, 2015.

[14] P. Ekman and H. Oster. Facial expressions of emotion. *Annual review of psychology*, 30(1):527–554, 1979.

[15] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein. Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation. *Proc. ACM SIGGRAPH*, 2018.

[16] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Proc. CVPR*, 2017.

[17] C. Gan, B. Gong, K. Liu, H. Su, and L. J. Guibas. Geometry guided convolutional neural networks for self-supervised video representation learning. In *Proc. CVPR*, 2018.

[18] F. C. Gerull and R. M. Rapee. Mother knows best: effects of maternal modelling on the acquisition of fear and avoidance behaviour in toddlers. *Behaviour research and therapy*, 40(3):279–287, 2002.

[19] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018.

[20] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proc. ICML*, 2015.

[21] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi. Conditional image generation for learning the structure of visual objects. *arXiv preprint arXiv:1806.07823*, 2018.

[22] X. Jia, B. De Brabandere, T. Tuytelaars, and L. V. Gool. Dynamic filter networks. In *NIPS*, 2016.

[23] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *Proc. ICLR*, 2015.

[24] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. In *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[25] T. D. Kulkarni, W. F. Whitney, P. Kohli, and J. Tenenbaum. Deep convolutional inverse graphics network. In *NIPS*, 2015.

[26] A. Kumar, A. Alavi, and R. Chellappa. KEPLER: keypoint and pose estimation of unconstrained faces by learning efficient H-CNN regressors. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2017.

[27] M. P. Kumar, B. Packer, and D. Koller. Self-paced learning for latent variable models. In *NIPS*. 2010.

[28] N. Kumar, A. Berg, P. Belhumeur, and S. Nayar. Describable visual attributes for face verification and image search. *IEEE PAMI*, 33(10):1962–1977, 2011.

[29] D. Lee, H. Park, and C. D. Yoo. Face alignment using cascade gaussian process regression trees. In *Proc. CVPR*, 2015.

[30] H.-Y. Lee, J.-B. Huang, M. Singh, and M.-H. Yang. Unsupervised representation learning by sorting sequences. In *Proc. ICCV*, 2017.

[31] Y. Li, R. Wang, H. Liu, H. Jiang, S. Shan, and X. Chen. Two birds, one stone: Jointly learning binary code for large-scale face image retrieval and attributes prediction. In *Proc. ICCV*, 2015.

[32] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proc. ICCV*, 2015.

[33] I. Loshchilov and F. Hutter. Online batch selection for faster training of neural networks. *ICLR Workshops*, 2016.

[34] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: unsupervised learning using temporal order verification. In *Proc. ECCV*, 2016.

[35] A. Mollahosseini, B. Hasani, and M. H. Mahoor. AffectNet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 2017.

[36] A. Nagrani, J. S. Chung, and A. Zisserman. VoxCeleb: a large-scale speaker identification dataset. In *INTERSPEECH*, 2017.

[37] A. Nagrani, S. Albanie, and A. Zisserman. Learnable pins: Cross-modal embeddings for person identity. *arXiv preprint arXiv:1805.00833*, 2018.

[38] M. Noroozi and P. Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proc. ECCV*, 2016.

[39] D. Novotny, D. Larlus, and A. Vedaldi. Anchornet: A weakly supervised network to learn geometry-sensitive features for semantic matching. In *Proc. CVPR*, 2017.

[40] D. Novotny, S. Albanie, D. Larlus, and A. Vedaldi. Self-supervised learning of geometrically stable features through probabilistic introspection. In *Proc. CVPR*, 2018.

[41] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *Proc. BMVC.*, 2015.

[42] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

[43] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, 2016.

[44] V. Pătrăucean, A. Handa, and R. Cipolla. Spatio-temporal video autoencoder with differentiable memory. In *NIPS*, 2016.

[45] S. Ren, X. Cao, Y. Wei, and J. Sun. Face alignment at 3000 fps via regressing local binary features. In *Proc. CVPR*, 2014.

[46] E. M. Rudd, M. Günther, and T. E. Boult. Moon: A mixed objective optimization network for the recognition of facial attributes. In *Proc. ECCV*, 2016.

[47] C. Sagonas, E. Antonakos, G. Tzimiropoulos, S. Zafeiriou, and M. Pantic. 300 faces in-the-wild challenge: Database and results. *Image and Vision Computing*, 47:3–18, 2016.

[48] A. Shrivastava, A. Gupta, and R. Girshick. Training region-based object detectors with online hard example mining. In *Proc. CVPR*, 2016.

[49] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proc. ICCV*, 2015.

[50] A. Tewari, M. Zollhöfer, H. Kim, P. Garrido, F. Bernard, P. Perez, and C. Theobalt. Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proc. ICCV*, 2017.

[51] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *Proc. ICCV*, 2017.

[52] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. In *NIPS*, 2017.

[53] L. Tran, X. Yin, and X. Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proc. CVPR*, 2017.

[54] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *Proc. ECCV*, 2018.

[55] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, 2015.

[56] O. Wiles, A. S. Koepke, and A. Zisserman. X2Face: A network for controlling face generation using images, audio, and pose codes. In *Proc. ECCV*, 2018.

[57] S. Xiao, J. Feng, J. Xing, H. Lai, S. Yan, and A. Kassim. Robust facial landmark detection via recurrent attentive-refinement networks. In *Proc. ECCV*, 2016.

[58] T. Xue, J. Wu, K. Bouman, and B. Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016.

[59] J. Yang, P. Ren, D. Chen, F. Wen, H. Li, and G. Hua. Neural aggregation network for video face recognition. In *Proc. CVPR*, 2017.

[60] X. Yu, F. Zhou, and M. Chandraker. Deep deformation network for object landmark localization. In *Proc. ECCV*, 2016.

[61] A. R. Zamir, T. Wekel, P. Agrawal, C. Wei, J. Malik, and S. Savarese. Generic 3d representation via pose estimation and matching. In *Proc. ECCV*, 2016.

[62] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *Proc. ECCV*, 2016.

[63] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. CVPR*, 2017.

[64] Y. Zhang, Y. Guo, Y. Jin, Y. Luo, Z. He, and H. Lee. Unsupervised discovery of object landmarks as structural representations. In *Proc. CVPR*, 2018.

[65] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Proc. ECCV*, 2014.

[66] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Learning deep representation for face alignment with auxiliary attributes. *IEEE PAMI*, 2016.

[67] E. Zhou, H. Fan, Z. Cao, Y. Jiang, and Q. Yin. Extensive facial landmark localization with coarse-to-fine convolutional network cascade. In *ICCVW*, 2013.

[68] T. Zhou, S. Tulsiani, W. Sun, J. Malik, and A. A. Efros. View synthesis by appearance flow. In *Proc. ECCV*, 2016.

[69] S. Zhu, C. Li, C. C. Loy, and X. Tang. Face alignment by coarse-to-fine shape searching. In *Proc. CVPR*, 2015.

[70] X. Zhu and D. Ramanan. Face detection, pose estimation and landmark localization in the wild. *Proc. CVPR*, 2012.

Supplementary Material: Self-supervised learning of a facial attribute embedding from video

Olivia Wiles*
ow@robots.ox.ac.uk
A. Sophia Koepke*
koepke@robots.ox.ac.uk
Andrew Zisserman
az@robots.ox.ac.uk

Visual Geometry Group
University of Oxford
Oxford, UK

We provide additional details about FAb-Net's architecture and training in Section 1, the self-supervised baselines in Section 2, pre-processing of the datasets in Section 3 and additional qualitative results in Section 4.

1 Additional details on architectures and training

Section 1.1 and Section 1.2 give additional details about the encoder-decoder architecture used and about the training respectively.

1.1 Architecture

The architecture of the encoders and decoders is based on pix2pix [12] but without skip-connections (see Fig. 1). It consists of encoders (with shared weights) and corresponding symmetrical decoders (also with shared weights). The face embedding vectors which are the outputs of the encoder have channel size 256. At the centre of the network, the source embedding vectors are concatenated pairwise with the target embedding vector. This 512-channel vector serves as input to the decoders.

The sampler decoders predict a $2 \times 256 \times 256$ output which determines how to sample from the source frame. When using multiple source frames, the network is augmented with confidence map decoders whose architecture is identical to the sampler decoders, except that the last layer outputs a 1-channel image. The confidence decoders also have shared weights for the different source frames.

1.2 Training and data augmentation

The images are first scaled to size 256×256. Because VoxCeleb1/VoxCeleb2 have different crops, VoxCeleb2 is re-cropped by padding the images by $[20, 80, 20, 30]$ pixels to the left, top, right, and bottom respectively and then taking a centre crop of size 190×190. Given the re-cropped VoxCeleb2 images and the VoxCeleb1 images, the images are augmented by

© 2018. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.

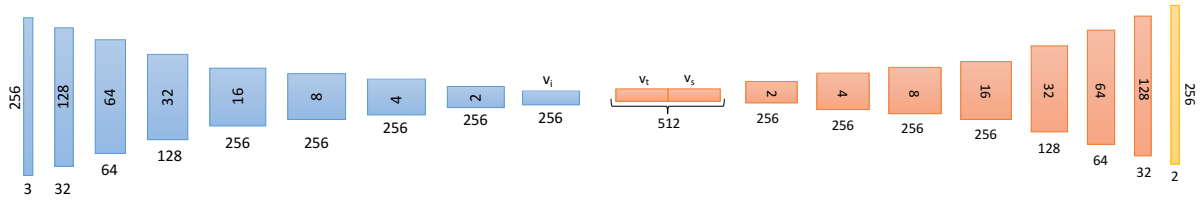


Figure 1: *Encoder-decoder*: In the encoder, each convolution is followed by a leaky ReLU (factor 0.2) and a batch-norm layer (except for the first which has no batch-norm); the convolutional filter sizes are 4×4 and the stride/padding is $2/1$. The face embedding vector has channel size 256. In the decoder, the face embedding vectors corresponding to the source/target frame (v_s, v_t) are concatenated giving a 512-dimensional embedding vector. In all decoder layers, the sequence of executions is: ReLU, bilinear upsampling, batch-norm. The convolutional filter sizes in the sampler decoder and the confidence decoder are 3×3, the stride/padding is $1/1$. The final $2 \times 256 \times 256$ result (or $1 \times 256 \times 256$ for the confidence decoders) is passed through a tanh layer to give FAb-Net's prediction for how to sample from the source frame.

taking random square crops with width in the range $[170, 190]$ for VoxCeleb2 and $[180, 200]$ for VoxCeleb1.

The models are trained using SGD with learning rate 0.001, momentum 0.9 and batch size $N = 8$ (unless the curriculum strategy for FAb-Net is used in which case $N = 32$). The learning rate is divided by a factor of 10 when the loss plateaus (unless the curriculum strategy for FAb-Net is used, in which case this occurs only after the curriculum strategy terminates).

2 Self-supervised baselines

FAb-Net is compared to an autoencoder and to two state-of-the-art self-supervised methods [3, 10]. FAb-Net's architecture but the appropriate loss functions and setups are used. More detail is given below. The baselines are trained with the same training hyperparameters and data augmentation as FAb-Net (please refer to Section 1.2).

Autoencoder. FAb-Net's encoder-decoder architecture are used. The autoencoder is trained to recreate the source frame via a 256-dimensional bottleneck vector, which is used later for evaluation. (The 256-dimensional vector output from the encoder serves as input to the decoder.)

Gidaris et al. [3]. Gidaris et al. apply a rotation $\theta \in \{0°, 90°, 180°, 270°\}$ to an image and train a CNN to predict the rotation that has been applied. To implement this baseline, FAb-Net's encoder (illustrated in Fig. 1) is used. A linear layer (with input channel size 256 and output channel size 4) is appended followed by a softmax layer. The network is trained with a cross-entropy loss. The 256-dimensional embedding is used for evaluation.

Zhang et al. [10]. Zhang et al. split an input image into L and ab channels (e.g. grey and colour channels) and then learn to reconstruct the ab channels from the L channel and the L channel from the ab channels. To do this, they effectively split the network into two smaller networks, each with half the capacity of the original network. To implement this baseline,

the exact same encoder-decoder structure as FAb-Net is used, except that it is divided into two subnetworks, each with half the capacity of the original network at each layer (e.g. 16 channels in the first layer and 128 channels in the embedding for each sub-network). The MSE loss is used as the reconstruction loss. The concatenation of the two 128-dimensional embeddings gives a 256-dimensional embedding, which is used for evaluation.

3 Datasets

For our models and baselines the input image size is 256×256 except for the VGG-Face descriptor which requires input images of size 224×224.

300-W and MAFL. FAb-Net is evaluated on 300-W and MAFL using the procedure outlined in [8]. In order to make the images more similar to those of VoxCeleb+, the images are re-cropped to make a tighter crop around the face region (this is a fair comparison as [8, 9, 11] re-crop to make the images more similar to those in CelebA [6] which they use as their training set).

AFLW. The images in AFLW [5] are resized to 256×256 for our models and to 224×224 for the evaluation of the VGG-Face descriptor.

EmotioNet. For EmotioNet [1], the classifier is trained on the subset of the dataset that is automatically annotated. We divide this subset into a training and a validation set (with a 80/20 split). The faces are detected using dlib [4] and cropped to 256×256. This results in $743,033$ images for training and $185,759$ images for validation. The independent set of about 25k images is used as test set. After detecting faces, this gives $25,517$ images for testing. This evaluation is done for the 11 AUs used in track 1 of the EmotioNet challenge 2017 [2].

AffectNet. For our experiments on AffectNet [7], we use the manually annotated subset of the dataset. As the AffectNet test set is not released, we use the released validation set to test on and randomly divide the training set into a training and a validation subset (with a 85/15 split). The faces are detected using dlib [4] and cropped to 256×256. Furthermore, images annotated as 'non-face', 'none' or 'uncertain' are discarded. This results in $287,055$ images for training, $57,411$ for validation and $3,989$ images for testing. The test set is balanced across the different emotion categories whereas the training data is not.

4 Additional qualitative results.

Additional examples of learned confidence heatmaps are visualised in Fig. 2 and additional examples for retrieval are visualised in Fig. 3.

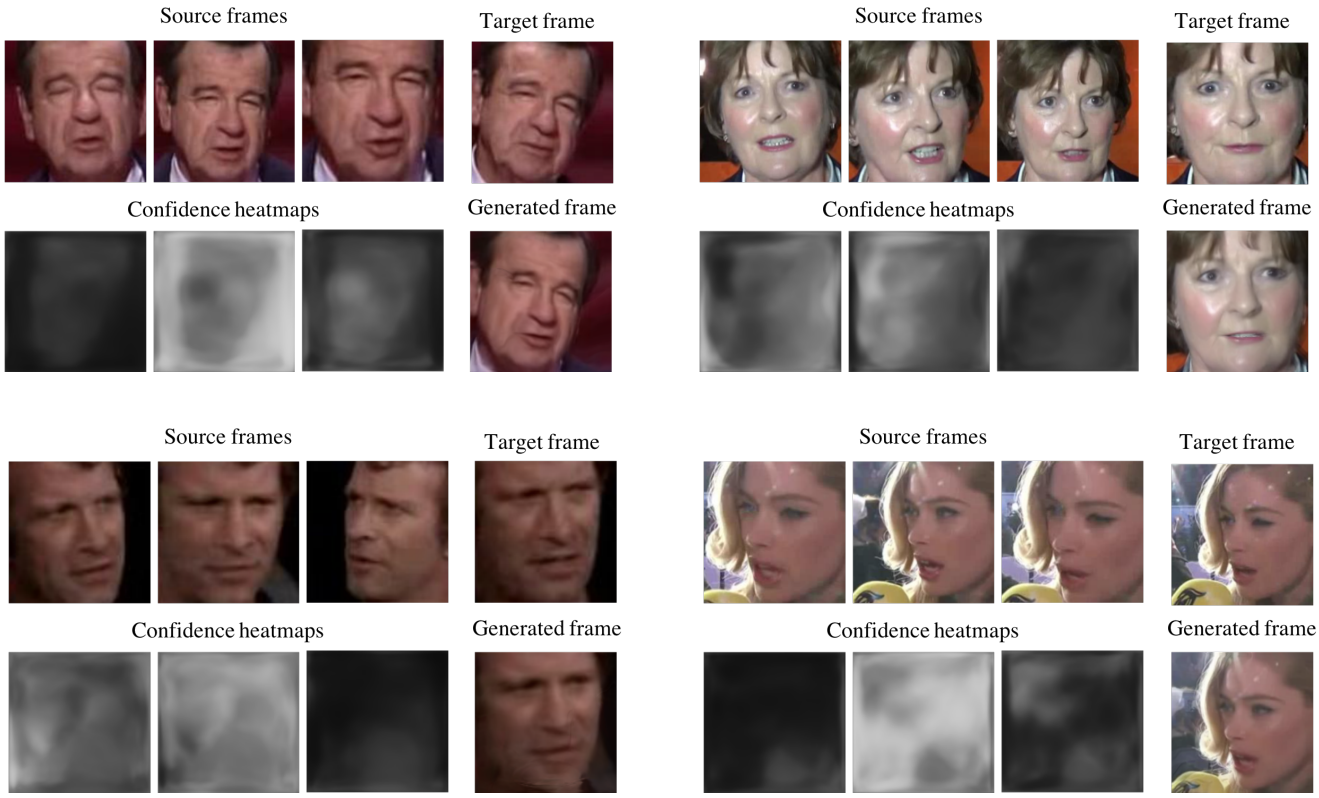


Figure 2: Additional examples of confidence heatmaps predicted by FAb-Net for the given source and target frames. These examples demonstrate how the confidence maps allow the network to focus on certain source frames or parts of different source frames. In the top left example, the network chooses one eye from the rightmost source frame even though its pose is quite different as compared to that of the target frame. In the bottom two examples, the source frames with pose or zoom dissimilar to that of the target frame are discarded.

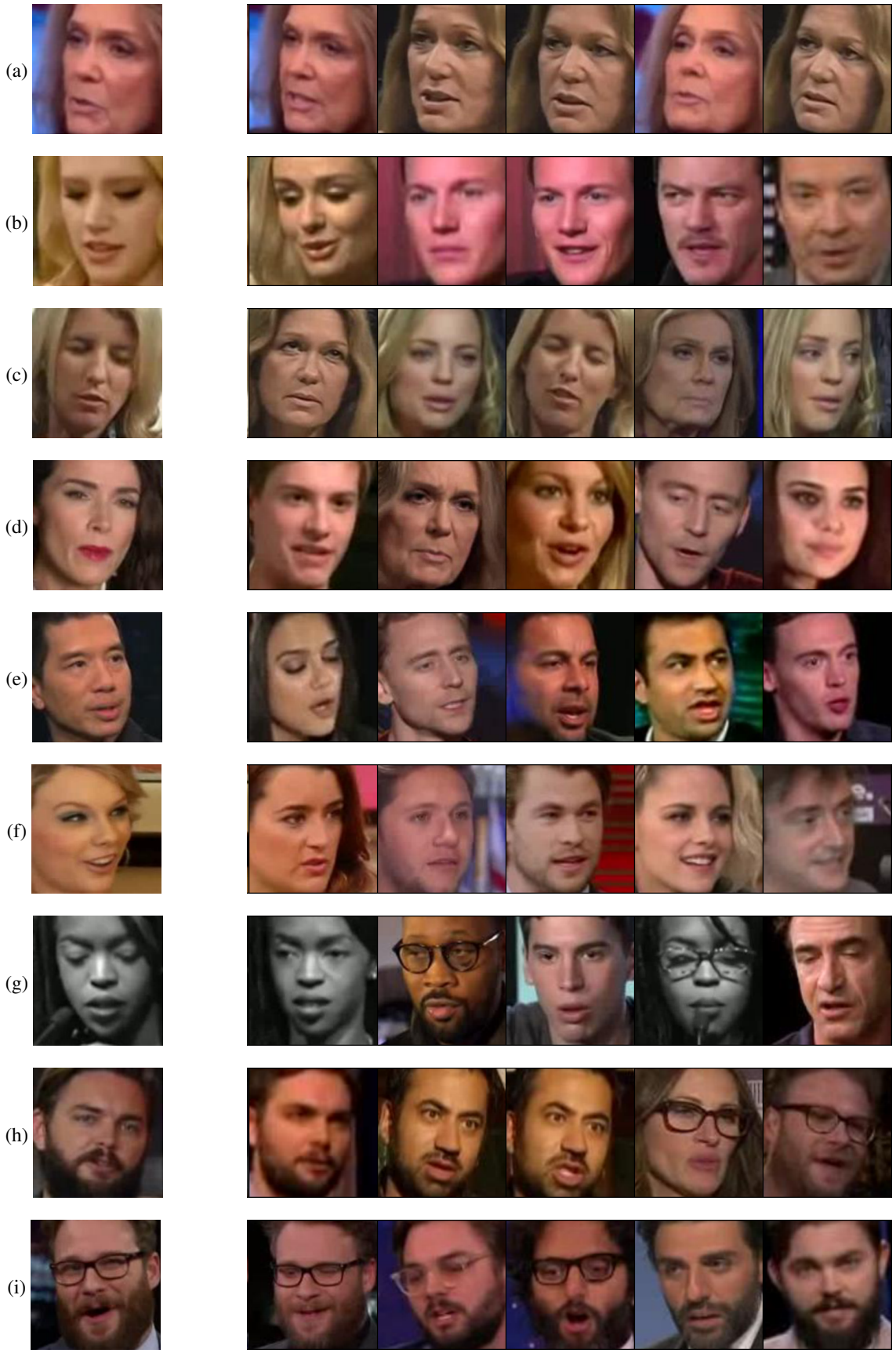


Figure 3: Additional results for nearest neighbour retrieval using the embedding learned by FAb-Net. The embedding captures facial attributes, e.g. pose and expression, as it succeeds in retrieving images with similar facial attributes given a query image.

References

[1] C. F. Benitez-Quiroz, R. Srinivasan, and A. M. Martinez. EmotioNet: An accurate, real-time algorithm for the automatic annotation of a million facial expressions in the wild. In *Proc. CVPR*, 2016.

[2] C. F. Benitez-Quiroz, R. Srinivasan, Q. Feng, Y. Wang, and A. M. Martinez. Emotionet challenge: Recognition of facial expressions of emotion in the wild. *arXiv preprint arXiv:1703.01210*, 2017.

[3] S. Gidaris, P. Singh, and N. Komodakis. Unsupervised representation learning by predicting image rotations. In *Proc. ICLR*, 2018.

[4] D. E. King. Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, 10:1755–1758, 2009.

[5] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof. Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. In *Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[6] Z. Liu, P. Luo, X. Wang, and X. Tang. Deep learning face attributes in the wild. In *Proc. ICCV*, 2015.

[7] A. Mollahosseini, B. Hasani, and M. H. Mahoor. AffectNet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, 2017.

[8] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *Proc. ICCV*, 2017.

[9] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. In *NIPS*, 2017.

[10] R. Zhang, P. Isola, and A. A. Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proc. CVPR*, 2017.

[11] Y. Zhang, Y. Guo, Y. Jin, Y. Luo, Z. He, and H. Lee. Unsupervised discovery of object landmarks as structural representations. In *Proc. CVPR*, 2018.

[12] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proc. ICCV*, 2017.

Chapter 7

Self-Supervised Learning of Class Embeddings from Video

This work was published at the ICCV workshop on Compact and Efficient Feature Representation and Learning in Computer Vision.

This work describes a self-supervised method for learning an embedding that encodes pose, expression, and 3D information specific to a class. It is a generalisation of the work in chapter 6 to more challenging domains and data by introducing a hierarchical photometric loss.

Contributions of Olivia Wiles

1. Worked on writing the paper
2. Developed the framework for testing the model on downstream tasks: in particular landmark estimation on the different classes
3. Developed ideas and wrote code for training the model (in particular the hierarchical model and how to handle multiple modes)
4. Ran experiments for the initial model and downstream tasks

Self-supervised learning of class embeddings from video

Olivia Wiles
University of Oxford
ow@robots.ox.ac.uk

A. Sophia Koepke
University of Oxford
koepke@robots.ox.ac.uk

Andrew Zisserman
University of Oxford
az@robots.ox.ac.uk

Abstract

This work explores how to use self-supervised learning on videos to learn a class-specific image embedding that encodes pose and shape information. At train time, two frames of the same video of an object class (e.g. human upper body) are extracted and each encoded to an embedding. Conditioned on these embeddings, the decoder network is tasked to transform one frame into another. To successfully perform long range transformations (e.g. a wrist lowered in one image should be mapped to the same wrist raised in another), we introduce a hierarchical probabilistic network decoder model. Once trained, the embedding can be used for a variety of downstream tasks and domains. We demonstrate our approach quantitatively on three distinct deformable object classes – human full bodies, upper bodies, faces – *and show experimentally that the learned embeddings do indeed generalise. They achieve state-of-the-art performance in comparison to other self-supervised methods trained on the same datasets, and approach the performance of fully supervised methods.*

1. Introduction

How much information is needed to learn a representation of an object class? Do we require separate representations for different aspects: e.g. one representation for 3D, another for pose, another for 2D landmarks? We investigate how to learn a single representation for a given object class that encodes multiple properties in a self-supervised manner. This representation can be used for further downstream tasks and domains with minimal additional effort.

We learn this representation – which we call an *image embedding* – in a self-supervised manner from a large collection of videos of that object class (e.g. human upper bodies, or talking heads). The principal assumption is that of *temporal coherence* – that frames of the video contain the object class, but *no* additional prior auxiliary information is required.

In order to learn the image embedding from a video dataset, the following proxy task is used. Given two frames from the same video, their image embeddings are used to warp one of the frames into the other.

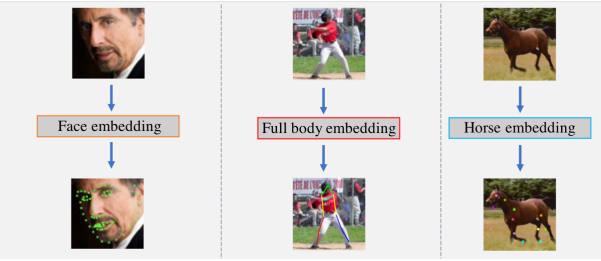


Figure 1. The aim of this work is to obtain a class-specific *image* embedding by self-supervised learning on a large collection of videos. The learned embedding can then be used for a variety of downstream tasks and datasets.

We want to model long range dependencies at high resolutions, for example large hand movements. In order to do this, we instantiate the warp probabilistically – for every pixel in one frame, we would like to predict the probability that that pixel corresponds to every other pixel in the other frame. Doing this naively is computationally prohibitive above a small (e.g. 32×32) resolution.

As a result, we use a hierarchical approach to perform this operation. The model first learns the probabilities at a low resolution, before refining the probabilities at successive layers while conditioning on the lower resolution predictions. While solving the proxy task at a small resolution may seem trivial, in fact low resolution images encode important salient information such as spatial layout and context [46]. This approach is inspired by the classical (i.e. pre deep learning) multi-resolution methods employed for optical flow and stereo matching [1, 4, 26, 29].

The embedding, trained using only pairs of video frames, is then used for the tasks of predicting landmarks and their visibility on a variety of datasets which may differ substantially from the initial dataset. Our paradigm is useful in applications, as it requires only one large network per class and one additional small network per down stream task.

In summary, our contributions are as follows.

1. A self-supervised class embedding (Section 3) that can

model complex large movements, e.g. the movement of arms or hands.

2. A hierarchical probabilistic network that allows us to estimate the probability that a given pixel in a given frame matches each pixel in another frame of the same video for high resolution images.

3. Two additional losses for learning this embedding. The confidence loss (Section 3.2) allows the model to express what portions of the target image can be reliably predicted from the source and what portions cannot. The cyclic loss (Section 3.3) enforces that the model does not degenerate into a trivial solution.

4. We demonstrate that the method learns a useful representation that can be used for downstream tasks on the same or different domains for a variety of object classes. Our method achieves state-of-the-art performance in comparison to other self-supervised methods trained on the same datasets. Finally, we show qualitative examples of using our approach for a non-human class, that of horses.

2. Related work

Here, we focus on self-supervised learning from video. We also cover class specific modelling, where a model of the object is extracted using auxiliary information and then applied to novel images.

Self-supervised learning on video collections. Learning from video [2, 10, 15, 17, 21, 22, 30, 31, 35, 40, 42, 47, 52, 62, 64] is a powerful paradigm, as unlike with image collections, there is additional temporal and sequential information. The aim of self-supervised learning from video can be to learn to predict future frames [47], or to learn to predict depth [12, 14, 62]. However, we are interested in learning a set of useful features (e.g. frame representations).

One approach is to use the temporal ordering or coherence as a proxy loss in order to learn the representation [10, 17, 22, 24, 30, 31, 49, 52, 64]. Other approaches use egomotion [2, 21] in order to enforce equivariance in feature space [21]. In contrast, [23] predicts the transformation applied to a spatio-temporal block. Instead of enforcing constraints on the features, one can learn features using a generative task of future or input frame prediction [15, 40, 42]. Another approach is to use colourisation to learn features and to track objects [48].

Unlike these works, our focus is to learn a feature representation for a specific class, which can be used to predict class-specific attributes. Most similar to our method is [53] which uses video to learn a representation of faces. However, they do not consider other object classes.

Self-supervised learning of landmarks. Instead of using proxy tasks to learn useful features, another line of self-supervised learning is to explicitly learn a set of landmarks. This can be done by conditioning image generation on the image landmarks [19, 60]. Another approach is to recover object structure by enforcing equivariance to image transformations [43, 44].

3. A self-supervised representation

This section introduces our self-supervised model and architecture (Fig. 2). The model is trained for the proxy task of transforming one frame into another frame in a hierarchical manner (Section 3.1). We allow the model to express uncertainty (Section 3.2) and use additional cyclic constraints (Section 3.3) to stop the learned transformation from degenerating. This gives the final training objective. We introduce the framework for the case of human upper bodies, but the same framework is used for the other classes considered in this paper (full human body, talking faces, horses).

3.1. Proxy task to train the network: Modelling the transformation between images

A source frame I_S and a target frame I_T are randomly selected from the same video. The proxy task to train the model consists of learning how to warp the source frame I_S into the target frame I_T. Both frames are mapped, using a convolutional encoder with shared weights, to image embeddings e_S and e_T respectively.

Conditioned on these embeddings, the model predicts the probability of a pixel in the generated frame I_G matching each pixel in I_S. These probabilities are used to generate the colour of a pixel by taking the weighted average. To introduce our notation, let $I_{S_{kl}}$ and $I_{T_{ij}}$ be the colours for pixel locations (k, l) and (i, j) in the source and target frame respectively. The network predicts the colour in the generated frame I_G at pixel location (i, j) as a linear combination of pixels in the source frame

$$I_{G_{ij}} = \sum_{k,l} \mathbf{A}_{ij,kl} I_{S_{kl}}, \tag{1}$$

where $\mathbf{A}_{ij,kl}$ is the probability that a pixel $I_{T_{ij}}$ in the target frame matches a pixel $I_{S_{kl}}$ in the source frame. We explicitly predict the match similarity $\mathbf{M}_{ij,kl}$ between a pixel $I_{S_{kl}}$ and $I_{T_{ij}}$ and normalise the $\mathbf{M}_{ij,kl}$ to give $\mathbf{A}_{ij,kl}$ (see Eqs. (3)-(5)). I_G should match the target frame I_T (Fig. 2a), which we enforce using a photometric L1 loss

$$\mathcal{L}_{ph} = |I_G - I_T|_1. \tag{2}$$

While using the naive weighted sum works for smaller resolution images, for larger images this becomes computationally prohibitive. To deal with this problem, we introduce our hierarchical approach (Fig. 2b). Learning in a

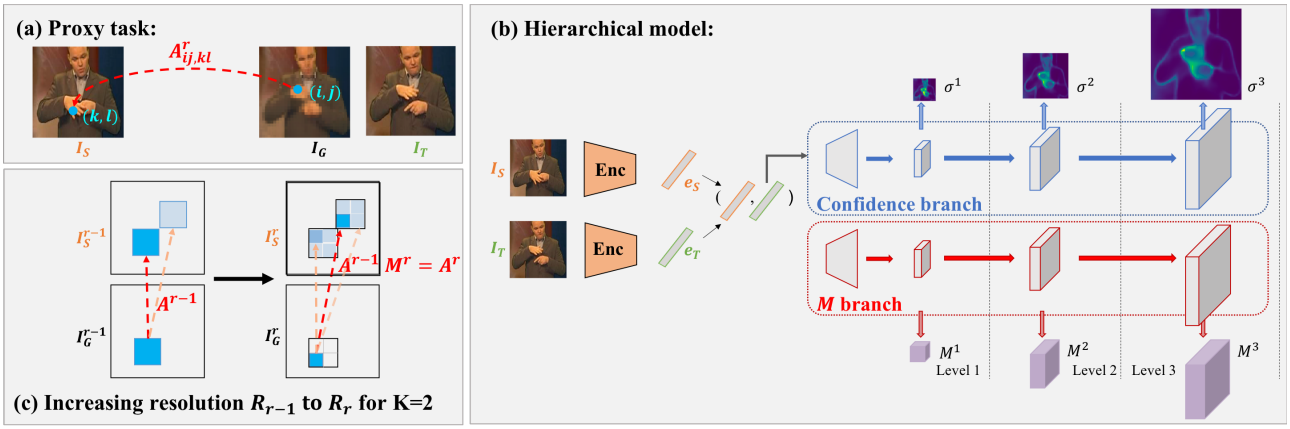


Figure 2. An overview of the approach. (a) The proxy task used to train the model. Given a source frame I_S and a target frame I_T, our model learns a mapping $\mathbf{A}^r$ to warp the source frame into a generated frame I_G. I_G should match I_T. (b) The model in more detail. The two frames are mapped to embeddings e_S, e_T. Conditioned on these embeddings, the model predicts the warp at an initial resolution $R_1 = (32 \times 32)$ as well as a confidence σ^1 for each pixel. These predictions are then refined at successively higher resolutions. (c) Illustration of how the predicted $\mathbf{M}^r$ at each resolution R_r are used to determine the warp $\mathbf{A}^r$.

hierarchical manner has been found to be useful in a number of tasks [9, 27, 28, 50, 51]. In our case, the network learns to determine roughly how to transform points (e.g. bigger parts of the image, like the arms) at a low resolution ($\mathbf{M}^1$ at level 1). This transformation is refined progressively at higher resolutions ($\mathbf{M}^r$ at level r). At these higher levels, the network can learn to focus on the details (e.g. the placement of the wrists). This can be regarded as a form of curriculum learning [3] where the decoder is progressively expanded in levels which increase the resolution of the generated image.

Probabilistic prediction at a low resolution (Training level 1). At the lowest resolution, $R_1 = (W_1 \times W_1) = (32 \times 32)$, we explicitly predict the probability $\mathbf{M}^1_{ij,kl}$ that each point (k, l) in the source frame matches each point (i, j) in the target frame. We then take the weighted average to obtain the probability distribution $\mathbf{A}^1_{ij,kl}$:

$$\mathbf{A}^1_{ij,kl} = \exp(\mathbf{M}^1_{ij,kl}) / \sum_{m,n} \exp(\mathbf{M}^1_{ij,mn}). \quad (3)$$

Using the computed probability distribution, we obtain the generated frame (Eq. (1)).

Refining the prediction at a higher resolution (Training level r.) Given the generated frame I^{r-1}_G at resolution R_{r-1}, we seek to refine I^{r-1}_G to obtain I^r_G at a higher resolution R_r. For a given location (i, j), the highest $\mathbf{A}^1_{ij,kl}$ give the most likely locations that (i, j) points to in the source frame. We will use this to limit the locations we consider at the higher resolution (see Fig. 2c).

In a traditional CNN, as we decode, we would have to keep track of the probabilities for a pixel (i, j) matching every pixel (k, l) in the source frame at that resolution. So doubling the resolution of the generated image at each layer requires quadrupling the number of predicted probabilities. Our insight is that keeping track of all of these probabilities is unnecessary. For a given pixel (i, j), we can throw away the unlikely matches at lower resolutions (effectively setting them to 0) while keeping track of the top K matches. Then when we double the resolution at the next layer, we only need to predict $4K$ values (if the width and height of the generated image has doubled, then one pixel at the lower level corresponds to four pixels at the higher level as illustrated in Fig. 2 c)). Instead of using these predicted $4K$ values as raw probabilities we use them to re-weight the probabilities predicted at the lower resolution to make the process differentiable. This leads to a sparser representation that grows quadratically.

The M-branch decoder is used to obtain the $4K$ values $\mathbf{M}^r$. These are multiplied by the probabilities at the lower resolution and a softmax normalisation is performed to obtain the final probability distribution $\mathbf{A}^r$:

$$\mathbf{P}^r_{ij,kl} = \mathbf{A}^{r-1}_{\lfloor \frac{i}{2} \rfloor \lfloor \frac{j}{2} \rfloor, \lfloor \frac{k}{2} \rfloor \lfloor \frac{l}{2} \rfloor} \mathbf{M}^r_{ij,kl} \quad (4)$$

$$\mathbf{A}^r_{ij,kl} = \exp(\mathbf{P}^r_{ij,kl}) / \sum_{m,n} \exp(\mathbf{P}^r_{ij,mn}). \quad (5)$$

Discussion. Our aim is to compute a cost volume that models the probability distribution of where a pixel in the target frame maps to in the source frame. [11] introduced using a cost volume in a deep learning framework for optical flow by computing the similarity between features. This idea has been leveraged in many recent works [45, 48]. However, naively comparing features at a $W \times W$ resolution requires computing W^4 values which quickly becomes prohibitively

large. As a result these methods are forced to use a small cost volume or a tiny batch size.

The grid sampler introduced in [18] provided another way to model the transformation between images by explicitly learning the warp field. This was used effectively by [53] in order to learn meaningful embeddings for faces. However, gradients only occur in the local neighbourhood of a point. As a result if the point needs to travel a large distance between images and there is no smooth colour transition (as is common in most images), then these gradients will be useless and the model will fail to learn.

Our hierarchical approach gives a way to address the limitation of both approaches. We can grow the cost volume to image resolutions of the same size as the original image with minimal overhead. We additionally do not suffer from the problem of local gradients. Finally, the hierarchical approach enforces the spatial constraint – that pixels in a local neighbourhood move together.

3.2. Modelling occlusion and background

When modelling the transformation between frames it is possible for part of an object to become occluded (e.g. the hand moving in front of the face) or un-occluded. Additionally, there may be parts of the scene that are not visible in the previous frame (e.g. for the signing videos the background is a video itself and constantly changing).

To allow the model to express uncertainty due to these challenges, we use an additional decoder which explicitly models the confidence σ^r at resolution R_r for the transformation at each location in I_G. Following [33], we assume that the pixel-wise confidence measure is Laplace distributed and use it to reweigh the photometric loss $\mathcal{L}^r_{ph}$ at each pixel:

$$\mathcal{L}^r_{con} = \sum_{i,j} -\ln \frac{\sqrt{2}}{2\sigma^r_{ij}} \exp\left(-\frac{\sqrt{2}|I_{G_{ij}} - I_{T_{ij}}|_1}{\sigma^r_{ij}}\right). \quad (6)$$

3.3. Dealing with multiple modes

One of the degeneracies that can occur when using the probabilistic approach is a non-injective mapping due to multiple colour modes (e.g. the three skin regions – two hands and the head). For example, a point on the left hand can be mapped to either hand or the head; the model is not forced to choose correctly between them. In practice, the model cheats and maps all these modes to the one that moves the least (the head).

The key idea here is to use a cyclic loss [41, 63] and normalisation to enforce uniqueness in order to avoid this problem. If pixels are transformed from I^1_S to I^1_T and back to I^1_S, then they should end up at their original location. If they do not, then it means multiple points in one image are mapped to the same point in another.

The cyclic loss enforces that pixels should return to their original location. It is formulated as the log likelihood of the expectation that a point in the source frame will end up back at the same point at level 1 of the hierarchical model,

$$\mathcal{L}_{cyc} = \frac{\sum_{kl} -\ln(\sum_{ij}(A^1_{kl,ij}A^1_{ij,kl}))}{W_1 W_1}. \quad (7)$$

The loss is minimised when each pixel (k, l) in the source frame maps with probability 1 to a point in the target frame and that same point in the target maps with probability 1 to the original point in the source, i.e. when $A^1_{ij,kl} = A^1_{kl,ij} = 1$.

To enforce uniqueness of the pixel transformation (e.g. that not all points in the source frame are mapped to the same point in the target), we perform a normalisation step before applying the cyclic loss. Points that map to many others in either the source or the target frame are down-weighted to give $A^1_{ij,kl}$:

$$A^1_{ij,kl} = \min\left(\frac{\mathbf{A}^1_{ij,kl}}{\sum_{m,n}\mathbf{A}^1_{mn,kl}}, \frac{\mathbf{A}^1_{ij,kl}}{\sum_{m,n}\mathbf{A}^1_{ij,mn}}\right). \quad (8)$$

The matches that still have a high probability are unique in both target and source, as required.

4. Architecture and training

All self-supervised models are trained using 3 levels with the lowest resolution $R_1 = (32 \times 32)$ which is increased to resolution $R_3 = (128 \times 128)$ (as we found additional levels led to marginal improvements). They are trained with $K = 9$, $\lambda = 1$, a learning rate of 0.001 and the Adam optimizer [25]. When sampling frame pairs from the video, we sample within a distance of 50 frames from the initial frame for upper body and horses, 20 frames for full human body and the whole face track for faces.

Architecture. We use a convolutional architecture similar to that of [53]. A 256×256 image is passed through 8 convolutional layers (interleaved with leaky ReLUs and batch-normalization) to give a $256D$ embedding. The confidence and M-decoder branches have the same structure but different weights. The concatenated embeddings are passed through 7 upsampling layers (composed of a ReLU, bilinear upsampler, convolution and batch-norm) to give a 128×128 resolution result. The intermediary outputs (e.g. $\mathbf{M}^r, \sigma^r$) are obtained by taking the feature map of resolution R_r and performing a 5×5 convolution to compress the number of channels.

Curriculum training strategy. The final training objective is the sum of the confidence loss at all layers and the cyclic loss weighted by a hyperparameter λ, $\mathcal{L} = \sum_i \mathcal{L}^r_{con} + \lambda\mathcal{L}_{cyc}$.

These losses are trained in a curriculum strategy. As the predictions of the higher layers depend on those of the

lower layers, we train the lower layers to a good local minimum before training the higher layers. We start at the lowest resolution R_1 and incorporate new layers when the loss plateaus. The model can first learn a rough estimation of how to transform the source frame into the target before iteratively refining at successively higher resolutions.

5. Experiments

We apply the learning framework of Section 3 to three distinct human object classes – *upper bodies*, *faces*, and *full human bodies* – to demonstrate its utility by modelling a variety of classes with different challenges. In addition to that, we show that our framework is useful for other, non-human object classes by presenting qualitative results for horses. The question we are seeking to answer here is whether the embedding that we learn from a large set of videos for each object class has encoded useful information about pose and shape of the object.

Downstream learning setup. Given an embedding learnt using self-supervision on one of the large video datasets, a regressor is trained to map this embedding to the downstream task (e.g. landmark prediction). This regressor is trained and then evaluated on the given train and test sets of the given dataset. For the regressor we consider a linear layer or a multi-layer perceptron containing two layers. While our embedding should learn about pose and expression, there is no reason to expect that the explicit landmarks should be linearly related to the embedding (this is unlike [19], which explicitly encode landmarks in their latent representation). Note that we are *not* training our encoder/embedding but *only* this regressor.

Training datasets. The upper body embedding is trained on the Extended BBC Pose dataset [5, 37] of people signing. The face embedding is trained on the VoxCeleb2 dataset [7] consisting of faces of people being interviewed. The full body embedding is trained on the Penn Action dataset [59] of people performing sporting actions. The horse embedding is trained on the horse subset of the TigDog dataset [8]. As our task is not to perform the detection but to learn a representation of the object class, we use the crops provided by the dataset or, if this is not available, a rough crop based on the provided information.

Baselines. We compare to two baselines. The first is using our encoder with random weights; this baseline shows how well our self-supervised training improves over random initialisation. The second baseline is [53] which uses a similar proxy task and capacity but a different loss function/architecture to learn the image embedding and a bilinear sampling for the transformation. They do not use a hierarchical approach or confidence predictions. We retrain [53] on upper body pose and fully body pose datasets using the authors' code provided online.

Other methods. We also report the results of other self-supervised and supervised methods on these datasets. These approaches vary in terms of how they pre-process their training data and assumptions made about the downstream task. We give these numbers to benchmark our approach against recent progress but note that these setups are not precisely the same.

5.1. Predicting landmarks

We consider the downstream task of predicting landmarks from our learnt embedding.

Evaluation metric. In order to evaluate the landmarks on upper body and full human body, we use the PCK metric [57]. This metric reports the percentage of correct keypoints within a normalised distance of the ground truth. The normalised distance depends on the dataset. In the case of BBC Pose, we use $d = 6$ pixels as is customary on this dataset. For FLIC we use a threshold of 0.2α where α is the torso diameter [38]. For Penn Action we use a threshold of $0.2 \max(s_w, s_h)$ where s_w, s_h are the width and height of the bounding box. For faces, we report the root mean squared error normalised by the interocular distance.

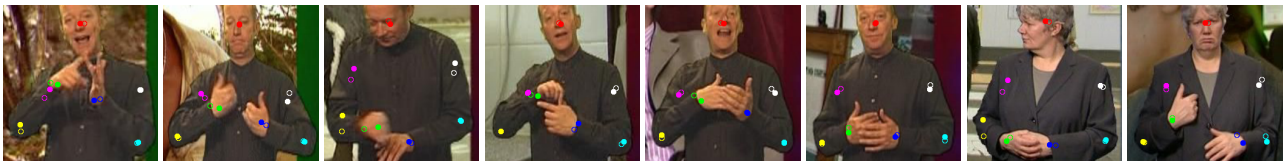
5.1.1 Upper body

We use the embedding trained on the BBC Pose dataset to predict upper body landmarks on the same dataset and on the FLIC dataset [38]. Quantitative results are discussed below, and qualitative results are shown in Fig. 3.

BBC Pose. The results on BBC Pose are given in Table 1. We first ablate our approach, demonstrating the utility of predicting confidences, and of using the cyclic loss $\mathcal{L}_{cyc}$. Each addition improves the average results and the results on the most challenging joint, the wrists. Using three levels as opposed to one improves performance, demonstrating the utility of the hierarchical approach.

In comparison to other self-supervised methods, our approach exhibits strong performance. It performs better than the baseline methods and [19], which was engineered to extract landmarks. [53] fails on this dataset due to the problem of local gradients – the movement between frames (e.g. of the hand) during training is too large, and it degenerates to predicting the identity transformation. Our approach is also better or competitive with most of the supervised methods. Clearly our embedding has indeed learned a semantically meaningful representation.

FLIC. Given that our approach outperforms the state-of-the-art on the BBC Pose dataset, we consider how well the embedding generalises to a new domain, the FLIC dataset, which consists of the upper body of people in film. The background and people are very different from the BBC



(a) **BBC**. Filled dots are GT, empty predictions.



(b) **FLIC**. Predicted poses.

Figure 3. Qualitative results on the upper body pose datasets. More examples are given in the supplementary material.

Table 1. Upper body landmark prediction on BBC Pose. Results reported are the PCK for $d < 6$. Higher is better. † denotes training with Extended BBC Pose, else with BBC Pose. The column *Loss* specifies the training losses used, $\mathcal{L}_{ph}$(ph), $\mathcal{L}_{cyc}$ (cyc) and $\mathcal{L}^{r}_{con}$ (con). r denotes the level/resolution at which training is stopped. $r = 1$ corresponds to a generated image of size 32×32, $r = 3$ to a generated image of size 128×128.

Method	Loss	Rg.	Hd	Wrt	Elb	Shldr	Avg
Ours							
r=1†	ph,cyc,con	lin	93.7	35.8	72.3	81.6	67.7
r=1†	ph,cyc,con	2 lr	94.2	51.2	78.7	82.4	74.1
r=3	ph,cyc,con	lin	**98.0**	30.7	78.9	71.3	65.6
r=3	ph,cyc,con	2 lr	96.5	41.0	82.4	73.2	69.9
r=3†	ph	2 lr	94.3	54.1	79.1	83.2	75.3
r=3†	ph,con	2 lr	96.0	58.3	**83.5**	**83.7**	78.1
r=3†	ph,cyc,con	2 lr	96.8	**62.1**	82.1	82.8	**78.7**
Self-supervised							
FAb-Net [53]†		2 lr	73.8	21.8	64.7	61.	52.9
Rand. init†		2 lr	73.2	23.2	64.5	54.7	51.1
Jakab *et al.* [19]		lin	81.1	49.1	53.1	70.1	60.7
Supervised							
Yang and Ramanan [56]			63.4	53.7	49.2	46.1	51.6
Pfister *et al.* [37]			74.9	53.1	46.0	71.4	59.4
Chen and Yuille [6]			65.9	47.9	66.5	76.8	64.1
Charles *et al.* [5]			95.4	73.9	68.7	90.3	79.9
Pfister *et al.* [36]			98.0	88.5	77.1	93.5	88.0

Pose dataset. As can be seen in Table 2, our approach generalises well to this new domain, achieving high performance. Again, using three levels as opposed to one improves performance.

5.1.2 Faces

The second class we consider is faces. As this model is trained on VoxCeleb2, which has no annotated keypoints, we test the learned embedding by predicting landmarks on a variety of other datasets. This additionally tests the embedding's generalisability.

Our embedding is used to regress landmarks on the AFLW, 300-W, and MAFL datasets and results are reported in Table 3. For AFLW, we report results on the 5-always visible landmarks (AFLW5) as well as for all 21 landmarks (AFLW21). Qualitative results are shown in Fig. 4.

Table 2. Upper body landmark prediction at PCK0.2 (as defined in [32]) on FLIC using the embedding trained on Extended BBC Pose. Higher is better. †The entire model is fine-tuned on the FLIC dataset, whereas we regress *only* two layers from the embedding.

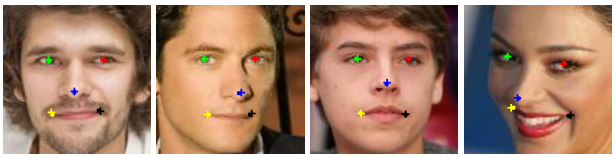
Method	Rg.	Hd	Shldr	Elb	Wrt	Avg
Ours						
r=1	2 lr	94.2	95.7	82.5	62.6	82.3
r=3	2 lr	97.2	97.1	84.8	65.2	84.5
Self-supervised						
Random init	2 lr	85.5	90.9	77.9	65.1	79.0
S&L [30]†		98.1	93.8	87.1	69.7	86.2
Supervised						
Newell *et al.* [32]		–	–	99.0	97.0	–

Our approach performs better than the baseline methods and other methods designed for predicting landmarks when trained with similar data. Our method even performs better than full frameworks trained (self-supervised or supervised) on the given dataset.

5.1.3 Full body

Finally, we test our method on full bodies using the Penn Action dataset [59]. The person may be seen from the front or back and performing a large variety of deformations which results in an extremely challenging dataset.

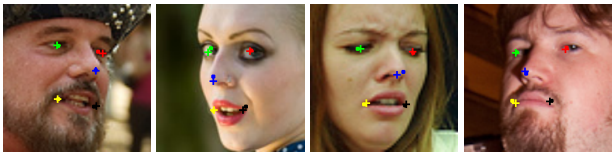
We use the learned embedding to regress landmarks. Quantitative results are reported in Table 4 and qualitative results in Fig. 5. We perform better than the baselines, and approach the performance of methods trained with deep learning on this dataset. Similarly to upper bodies, [53] degenerates to predicting the identity transformation, demonstrating the effectiveness of our method.



(a) **MAFL**. Crosses are predictions, dots GT.



(b) **300W**. Blue is GT, green predictions.



(c) **AFLW5**. Crosses are predictions, dots GT.

Figure 4. Qualitative results on the face datasets. More examples are given in the supplementary material.

Table 3. Face landmark prediction error on the 300-W and MAFL, AFLW datasets. Lower is better. † denotes trained on VoxCeleb 1/2, ‡ on VoxCeleb 1. Note that MAFL is a subset of CelebA and models trained on CelebA are fine-tuned on AFLW when reporting results on this dataset. Our embedding is never fine-tuned on these datasets; *only* the regressor is trained.

Method	Regr.	**300-W**	**MAFL**	**AFLW5**	**AFLW21**
Self-supervised					
Trained on VoxCeleb2					
Ours					
r=3	lin	4.93	3.21	6.73	**7.16**
r=1	2 lr	5.42	3.55	7.30	7.84
r=3	2 lr	**4.70**	**2.98**	**6.64**	7.28
FAb-Net [53]†	lin	5.71	3.44	7.52	8.08
Jakab *et al.* [19]‡	lin	–	3.63	6.75	–
Jakab *et al.* [20]‡	lin	5.37	–	–	–
Trained on CelebA					
Jakab *et al.* [19]	lin	–	**2.54**	**6.33**	–
Zhang *et al.* [60]	lin	–	3.16	6.58	–
Thewlis *et al.* [44]	lin	9.30	6.67	10.53	–
Thewlis *et al.* [43]	lin	7.97	5.83	8.80	–
Supervised					
MTCNN [61]		–	**5.39**	6.90	–
TCDCN [58]		5.54	–	7.65	–
RAR [54]		**4.94**	–	7.23	–

5.1.4 Non-human object classes: horses

A big advantage of our self-supervised framework is that we can get embeddings for any object class, provided we have video data to train with. To show this, we obtain a horse embedding by training on the horse subset of the TigDog dataset. We train a 2-layer regressor from the embedding to the provided keypoints. Example results can be seen in Fig. 6, more results are shown in the supplementary material.



Figure 5. Full body 2D landmarks results on the Penn Action dataset.

Table 4. Full body landmark prediction at PCK0.2 (as defined in [39]) on rhe Penn Action dataset. Higher is better.

Method	Regr.	Hd	Shldr	Elb	Wrt	Hip	Knee	Ankl	Mean
Ours									
r=1	2 lr	80.7	76.4	66.3	54.2	79.3	76.3	76.5	72.6
r=3	2 lr	**83.0**	**78.8**	**71.0**	**58.3**	**80.9**	**78.6**	**76.9**	**75.1**
Self-supervised									
FAb-Net [53]	2 lr	69.3	59.1	50.2	34.0	68.8	62.2	57.5	56.4
Random init	2 lr	70.5	60.4	50.4	35.1	70.9	63.5	53.9	56.8
Supervised									
Park and Ramanan [34]		62.8	52.0	32.3	23.3	53.3	50.2	43.0	45.3
Nie *et al.* [55]		64.2	55.4	33.8	24.4	56.4	54.1	48.0	48.0
Iqbal *et al.* [16]		89.1	86.4	73.9	72.0	85.3	79.0	80.3	81.1
Gkioxari *et al.* [13]		95.6	93.8	90.4	90.7	91.8	90.8	91.5	91.8
Song *et al.* [39]		97.6	96.8	95.2	95.1	97.0	96.8	96.9	96.4



Figure 6. 2D landmarks results on horses from the TigDog dataset.

5.2. Predicting visibility

While we have extensively investigated and demonstrated the high quality of the learned embedding by using it to regress landmarks, here we investigate whether the embedding has learned something beyond landmarks. In particular, we consider whether our embedding can be used to predict whether a landmark is or is not visible. Self-supervised methods for detecting landmarks, such as [19] cannot perform this task, as they explicitly use the landmarks in their representation.

Both the Penn Action and AFLW datasets have visibility annotations. We train a 2-layer multi-layer perceptron from the embedding to predict visibility for each landmark using a binary-cross entropy loss. We compute the area under the curve (AUC) and average over each landmark. For AFLW, we obtain 89.0 AUC and for Penn Action 77.4 AUC. A network with random initialisation achieves 63.3 AUC for PennAction and 76.6 for AFLW. This demonstrates that our method has learned something beyond just 2D positioning.

6. Conclusion

We have introduced a novel method for learning an embedding which encodes high-fidelity 2D landmarks using self-supervision on video. Because our method is self-supervised, we can incorporate an unlimited amount of data from varied domains to improve the learned embedding and only use a small set of training data in order to learn the mapping from the embedding to downstream tasks or domains. We explore further in the supplementary material how the downstream performance varies with the size of this downstream training set. We have demonstrated the method for four distinct deformable or articulated classes, but it is equally applicable to rigid classes (e.g. cars).

There are many interesting future directions. The embedding can be learnt for more animal classes and used for other downstream tasks. Also, the embedding could be extended to incorporate the temporal component implicit in the video in order to summarise multiple frames.

Acknowledgements

This work is supported by the EPSRC programme grant Seebibyte EP/M013774/1: Visual Search for the Era of Big Data.

References

[1] E. H. Adelson, C. H. Anderson, J. R. Bergen, P. J. Burt, and J. M. Ogden. Pyramid methods in image processing. *RCA engineer*, 1984. 1

[2] P. Agrawal, J. Carreira, and J. Malik. Learning to see by moving. In *Proc. ICCV*, 2015. 2

[3] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proc. ICML*, 2009. 3

[4] T. Brox, C. Bregler, and J. Malik. Large displacement optical flow. In *Proc. CVPR*, 2009. 1

[5] J. Charles, T. Pfister, D. Magee, D. Hogg, and A. Zisserman. Domain adaptation for upper body pose tracking in signed TV broadcasts. In *Proc. BMVC*, 2013. 5, 6

[6] X. Chen and A. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *NeurIPS*, 2014. 6

[7] J. S. Chung, A. Nagrani, and A. Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018. 5

[8] L. Del Pero, S. Ricco, R. Sukthankar, and V. Ferrari. Behavior discovery and alignment of articulated object classes from unstructured video. *IJCV*, 2016. 5

[9] E. L. Denton, S. Chintala, and R. Fergus. Deep generative image models using a laplacian pyramid of adversarial networks. In *NeurIPS*, 2015. 3

[10] B. Fernando, H. Bilen, E. Gavves, and S. Gould. Self-supervised video representation learning with odd-one-out networks. In *Proc. ICCV*, 2017. 2

[11] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. Van der Smagt, D. Cremers, and T. Brox. Flownet: Learning optical flow with convolutional networks. In *Proc. ICCV*, 2015. 3

[12] R. Garg, V. K. BG, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *Proc. ECCV*, 2016. 2

[13] G. Gkioxari, A. Toshev, and N. Jaitly. Chained predictions using convolutional neural networks. In *Proc. ECCV*, 2016. 7

[14] C. Godard, O. Mac Aodha, and G. J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proc. CVPR*, 2017. 2

[15] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proc. CVPR*, 2015. 2

[16] U. Iqbal, M. Garbade, and J. Gall. Pose for action-action for pose. In *Proc. Int. Conf. Autom. Face and Gesture Recog.*, 2017. 7

[17] P. Isola, D. Zoran, D. Krishnan, and E. H. Adelson. Learning visual groups from co-occurrences in space and time. In *Proc. ICLR*, 2015. 2

[18] M. Jaderberg, K. Simonyan, and A. Zisserman. Spatial transformer networks. In *NeurIPS*, 2015. 4

[19] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks through conditional image generation. In *NeurIPS*, 2018. 2, 5, 6, 7

[20] T. Jakab, A. Gupta, H. Bilen, and A. Vedaldi. Learning landmarks from unaligned data using image translation. *arXiv preprint arXiv:1907.02055*, 2019. 7

[21] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proc. ICCV*, 2015. 2

[22] D. Jayaraman and K. Grauman. Slow and steady feature analysis: higher order temporal coherence in video. In *Proc. CVPR*, 2016. 2

[23] L. Jing and Y. Tian. Self-supervised spatiotemporal feature learning by video geometric transformations. *arXiv preprint arXiv:1811.11387*, 2018. 2

[24] D. Kim, D. Cho, and I. S. Kweon. Self-supervised video representation learning with space-time cubic puzzles. In *AAAI*, 2018. 2

[25] D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proc. ICLR*, 2014. 4

[26] R. Koch. 3-d surface reconstruction from stereoscopic image sequences. In *Proc. ICCV*, 1995. 1

[27] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proc. CVPR*, 2017. 3

[28] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang. Fast and accurate image super-resolution with deep laplacian pyramid networks. *IEEE PAMI*, 2018. 3

[29] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. IJCAI*, 1981. 1

[30] I. Misra, C. L. Zitnick, and M. Hebert. Shuffle and learn: Unsupervised learning using temporal order verification. In *Proc. ECCV*, 2016. 2, 6

[31] H. Mobahi, R. Collobert, and J. Weston. Deep learning from temporal coherence in video. In *Proc. ICML*, 2009. 2

[32] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *Proc. ECCV*, 2016. 6

[33] D. Novotny, D. Larlus, and A. Vedaldi. Learning 3D object categories by looking around them. In *Proc. ICCV*, 2017. 4

[34] D. Park and D. Ramanan. N-best maximal decoders for part

models. In *Proc. ICCV*, 2011. 7

[35] D. Pathak, R. Girshick, P. Dollár, T. Darrell, and B. Hariharan. Learning features by watching objects move. In *Proc. CVPR*, 2017. 2

[36] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *Proc. ICCV*, 2015. 6

[37] T. Pfister, K. Simonyan, J. Charles, and A. Zisserman. Deep convolutional neural networks for efficient pose estimation in gesture videos. In *Proc. ACCV*, 2014. 5, 6

[38] B. Sapp and B. Taskar. Modec: Multimodal decomposable models for human pose estimation. In *Proc. CVPR*, 2013. 5

[39] J. Song, L. Wang, L. Van Gool, and O. Hilliges. Thin-slicing network: A deep structured model for pose estimation in videos. In *Proc. CVPR*, 2017. 7

[40] N. Srivastava, E. Mansimov, and R. Salakhudinov. Unsupervised learning of video representations using lstms. In *Proc. ICML*, 2015. 2

[41] N. Sundaram, T. Brox, and K. Keutzer. Dense point trajectories by gpu-accelerated large displacement optical flow. In *Proc. ECCV*, 2010. 4

[42] G. W. Taylor, R. Fergus, Y. LeCun, and C. Bregler. Convolutional learning of spatio-temporal features. In *Proc. ECCV*, pages 140–153, 2010. 2

[43] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object frames by dense equivariant image labelling. In *NeurIPS*, 2017. 2, 7

[44] J. Thewlis, H. Bilen, and A. Vedaldi. Unsupervised learning of object landmarks by factorized spatial embeddings. In *Proc. ICCV*, 2017. 2, 7

[45] J. Thewlis, S. Zheng, P. H. S. Torr, and A. Vedaldi. Fully-trainable deep matching. In *Proc. BMVC*, 2016. 3

[46] A. Torralba, R. Fergus, and W. T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE PAMI*, 2008. 1

[47] C. Vondrick, H. Pirsiavash, and A. Torralba. Anticipating visual representations from unlabeled video. In *Proc. CVPR*, 2016. 2

[48] C. Vondrick, A. Shrivastava, A. Fathi, S. Guadarrama, and K. Murphy. Tracking emerges by colorizing videos. In *Proc. ECCV*, 2018. 2, 3

[49] X. Wang and A. Gupta. Unsupervised learning of visual representations using videos. In *Proc. ICCV*, 2015. 2

[50] Y. Wang, F. Perazzi, B. McWilliams, A. Sorkine-Hornung, O. Sorkine-Hornung, and C. Schroers. A fully progressive approach to single-image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 3

[51] Y.-X. Wang, D. Ramanan, and M. Hebert. Growing a brain: Fine-tuning by increasing model capacity. In *Proc. CVPR*, 2017. 3

[52] D. Wei, J. Lim, A. Zisserman, and W. T. Freeman. Learning and using the arrow of time. In *Proc. CVPR*, 2018. 2

[53] O. Wiles, A. S. Koepke, and A. Zisserman. Self-supervised learning of a facial attribute embedding from video. In *Proc. BMVC*, 2018. 2, 4, 5, 6, 7

[54] S. Xiao, J. Feng, J. Xing, H. Lai, S. Yan, and A. Kassim. Robust facial landmark detection via recurrent attentive-refinement networks. In *Proc. ECCV*, 2016. 7

[55] B. Xiaohan Nie, C. Xiong, and S.-C. Zhu. Joint action recognition and pose estimation from video. In *Proc. CVPR*, 2015. 7

[56] Y. Yang and D. Ramanan. Articulated pose estimation with flexible mixtures-of-parts. In *Proc. CVPR*, 2011. 6

[57] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE PAMI*, 2013. 5

[58] R. Zhang, P. Isola, and A. A. Efros. Colorful image colorization. In *Proc. ECCV*, 2016. 7

[59] W. Zhang, M. Zhu, and K. Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proc. ICCV*, 2013. 5, 6

[60] Y. Zhang, Y. Guo, Y. Jin, Y. Luo, Z. He, and H. Lee. Unsupervised discovery of object landmarks as structural representations. In *Proc. CVPR*, 2018. 2, 7

[61] Z. Zhang, P. Luo, C. C. Loy, and X. Tang. Facial landmark detection by deep multi-task learning. In *Proc. ECCV*, 2014. 7

[62] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. Unsupervised learning of depth and ego-motion from video. In *Proc. CVPR*, 2017. 2

[63] T. Zhou, P. Krähenbühl, M. Aubry, Q. Huang, and A. A. Efros. Learning dense correspondence via 3D-guided cycle consistency. In *Proc. CVPR*, 2016. 4

[64] Y. Zhou and T. L. Berg. Temporal perception and prediction in ego-centric video. In *Proc. ICCV*, 2015. 2

Supplementary material:
Self-supervised learning of class embeddings from video

Table 1. Performance on BBCPose (an upper body dataset) as a function of percentage of data used.

% of Data	Hd	Wrt	Elb	Shldr	Avg
5	95.40	50.70	80.20	75.15	72.50
10	94.10	54.30	81.50	77.35	74.34
25	96.90	57.10	80.00	80.40	75.99
50	95.80	60.90	82.30	79.70	77.37
100	96.80	62.10	82.10	82.80	78.70

1. Overview

In this supplementary material, we report the results of using less training data for the downstream tasks to train our embedding in Section 2, provide additional qualitative results for predicting keypoints and visibility in Section 3, and additional architecture details in Section 4.

2. Performance as a function of data

In this section we look at the results as a function of the amount of data used. The question we seek to validate is whether we can use a fraction of the original training data for the downstream task and still obtain results comparable with using the entire training set. As we are regressing only a small network from the embedding to the downstream task, there should be less danger of over fitting. As a result, we should be able to use a fraction of the available training data for the downstream task and achieve comparable results to using all the available training data.

Upper body. In Table 1, we investigate the performance as a function of the percentage of training data used. Note that there is some noise in the training annotations. We find that using 50% of the data gives the same performance as using all, using 10% of the data to get a small drop in performance of 5% percent and training with 5% of the data to get a larger drop in performance of approximately 7.5%. This validates that with our embedding we can use less training data for the downstream task and achieve comparable results to using all.

Faces. In Fig. 1, we plot the error for 300W, MAFL, and AFLW5 as a function of the percentage of data used. The error reported is the distance between predicted and ground truth landmarks as a percentage of the interocular distance (IoD). We can use a twentieth of the data and suffer a degradation in performance of only approximately 2% in terms of IoD for MAFL and 5% for 300W and AFLW. These results validate that we can use less data for these downstream tasks and still achieve comparable results to using all.

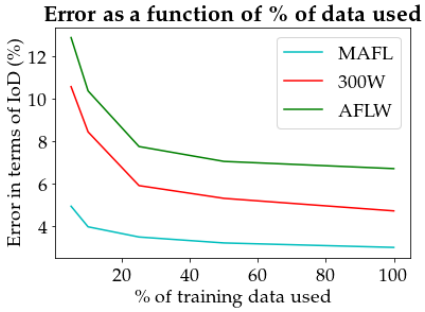


Figure 1. IoD on three face datasets – 300W, MAFL, and AFLW5 – as a function of the percentage of training data used.

3. Qualitative results for keypoints and visibility

Additional results of our model on BBCPose and FLIC are given in Fig. 2. Additional results of our model on AFLW5 and 300W are given in Fig. 3. Qualitative results of our model on AFLW21 with associated landmarks and visibility predictions are also given in Fig. 3. Additional results of our model applied to horses are given in Fig. 4. These results demonstrate that our model works on a variety of datasets.

4. Architecture

An overview of the architecture used including the encoder, confidence branch and M branch is given in Fig. 5.



(a) Additional results on BBCPose. A filled dot denotes ground truth, an empty one predictions.



(b) Additional results on FLIC.

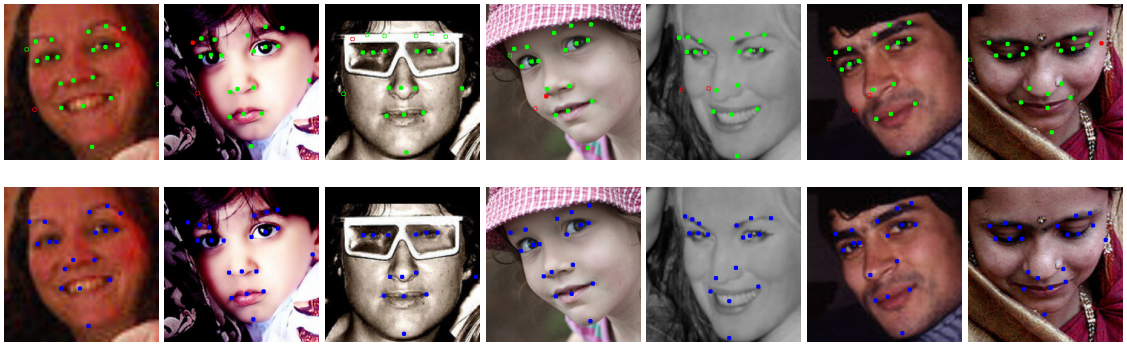
Figure 2. Additional results on pose datasets.



(a) Additional results on AFLW5. A cross denotes a prediction, a dot ground truth.



(b) Additional results on 300W. Top row shows predictions, bottom row ground truth.



(c) Additional results on AFLW21. Top row shows predictions, bottom row ground truth. For the predictions, green indicates our model predicted the point was visible, red invisible. A filled dot means that the point is actually visible according to the true annotation, an empty dot that it is invisible.

Figure 3. Additional results on faces.



Figure 4. Additional results on horses. Dots are predicted keypoints.

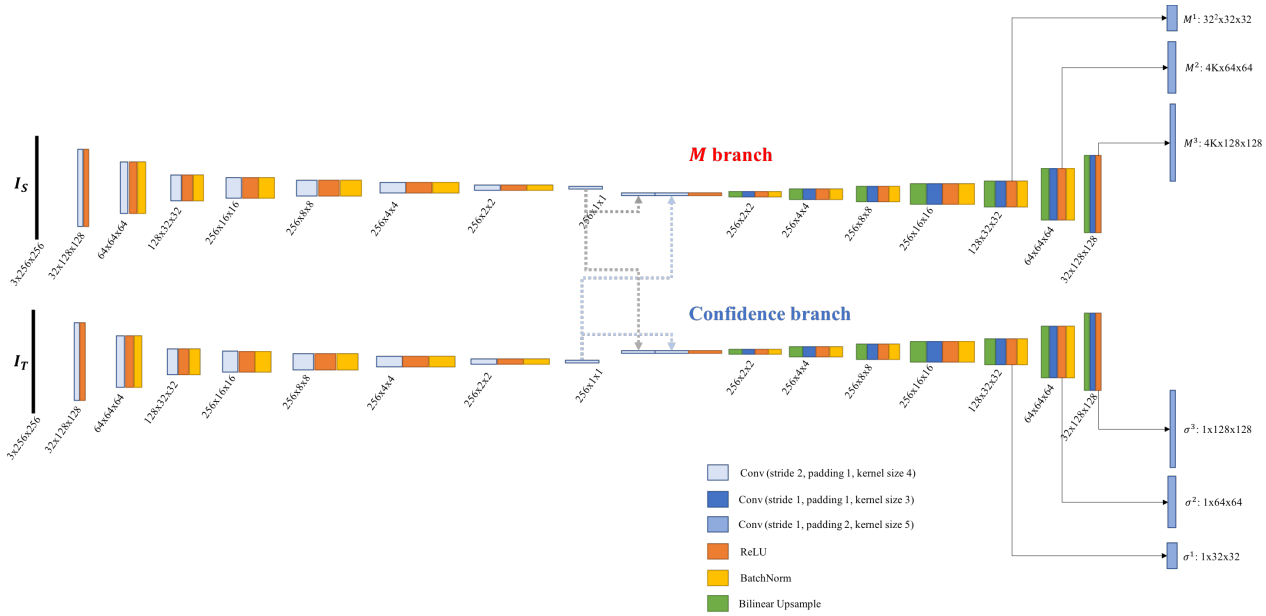


Figure 5. An overview of the entire architecture. Once we obtain the $256 \times 1 \times 1$ embeddings for two images, these are concatenated and fed into the two decoder branches. In order to obtain the confidence σ^r and $\mathbf{M}^r$ predictions at different resolutions, we extract feature maps at different resolutions from the decoder. These are then passed through a 5x5 convolution (indicated by arrows) to obtain an output with the required resolution and the required number of channels (e.g. 1 for σ^r). For the $\mathbf{M}$ branch, the prediction $\mathbf{M}^1$ at the lowest resolution is of size $1024 \times 32 \times 32$. This gives the matching probability between each pixel in the generated image ($32^2 = 1024$ pixels) and every pixel in the source image at the 32×32 resolution. These probabilities are refined at successive resolutions as follows. For each pixel location in the generated image, we only consider the top K matches (e.g. locations) at the previous resolution. Due to the 2x upsampling, this corresponds to $4K$ locations in the source image at the higher resolution. At $\mathbf{M}^2$, the prediction is then of size $4K \times 64 \times 64$ sorted from the top match to the top K^{th} match. Multiplying these values by the corresponding ones at the lower layer (as described in the main paper) gives the refined match scores at this resolution. The final probabilities $\mathbf{A}^r$ at this resolution are then obtained by normalising these refined match scores.

Chapter 8

X2Face: A Network for Controlling Face Generation using Images, Audio, and Pose Codes

This work was presented as a poster and published at ECCV 2018.

This is the first of the two chapters that introduce novel pipelines based on CNNs to solve view synthesis applications. This chapter describes a neural network for allowing manipulation of a face by another face, audio or pose information. The model automatically learns a canonicalisation network that puts the face to be manipulated into a canonical pose and a driving network that transforms this canonicalised face to the desired pose and expression of the other face. The model is trained in a self-supervised manner; the model is not explicitly trained to learn a canonical face. The trained model can then be used, as is, to manipulate a face using audio or a pose code by training a set of linear layers to transform the embedding within the driving network.

Contributions of Olivia Wiles

1. Developed the model and self-supervised pretext task as well as how to combine different views
2. Ran experiments for the initial model
3. Developed the method and initial framework for using the model for the downstream tasks including pose, audio, and the unwrapped mosaic
4. Worked on writing the paper

X2Face: A network for controlling face generation using images, audio, and pose codes

Olivia Wiles*, A. Sophia Koepke*, Andrew Zisserman

Visual Geometry Group,
University of Oxford
{ow,koepke,az}@robots.ox.ac.uk

Abstract. The objective of this paper is a neural network model that controls the pose and expression of a given face, using another face or modality (e.g. audio). This model can then be used for lightweight, sophisticated video and image editing.
We make the following three contributions. First, we introduce a network, X2Face, that can control a *source* face (specified by one or more frames) using another face in a *driving* frame to produce a *generated* frame with the identity of the *source* frame but the pose and expression of the face in the *driving* frame. Second, we propose a method for training the network fully self-supervised using a large collection of video data. Third, we show that the generation process can be driven by other modalities, such as audio or pose codes, without any further training of the network.
The generation results for driving a face with another face are compared to state-of-the-art self-supervised/supervised methods. We show that our approach is more robust than other methods, as it makes fewer assumptions about the input data. We also show examples of using our framework for video face editing.

1 Introduction

Being able to animate a still image of a face in a controllable, lightweight manner has many applications in image editing/enhancement and interactive systems (e.g. animating an on-screen agent with natural human poses/expressions). This is a challenging task, as it requires representing the face (e.g. modelling in 3D) in order to control it and a method of mapping the desired form of control (e.g. expression or pose) back onto the face representation. In this paper we investigate whether it is possible to forgo an explicit face representation and instead implicitly learn this in a self-supervised manner from a large collection of video data. Further, we investigate whether this implicit representation can then be used directly to control a face with another modality, such as audio or pose information.
To this end, we introduce X2Face, a novel self-supervised network architecture that can be used for face puppeteering of a *source* face given a *driving vector*.

* Denotes equal contribution.

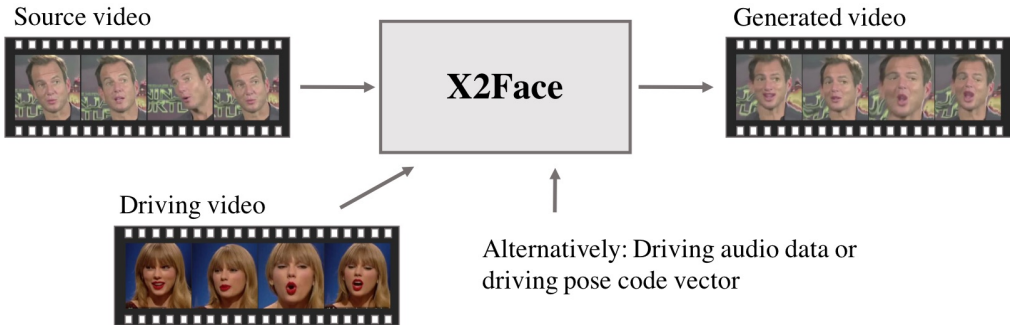


Fig. 1: Overview of X2Face: a model for controlling a *source* face using a *driving* frame, audio data, or specifying a pose vector. X2Face is trained without expression or pose labels.

The *source* face is instantiated from a single or multiple *source* frames, which are extracted from the same face track. The *driving vector* may come from multiple modalities: a *driving* frame from the same or another video face track, pose information, or audio information; this is illustrated in Fig. 1. The *generated* frame resulting from X2Face has the identity, hairstyle, etc. of the *source* face but the properties of the *driving vector* (e.g. the given pose, if pose information is given; or the *driving* frame's expression/pose, if a *driving* frame is given).
The network is trained in a self-supervised manner using pairs of *source* and *driving* frames. These frames are input to two subnetworks: the *embedding network* and the *driving network* (see Fig. 2). By controlling the information flow in the network architecture, the model learns to factorise the problem. The *embedding network* learns an *embedded* face representation for the *source* face – effectively face frontalisation; the *driving network* learns how to map from this *embedded* face representation to the *generated* frame via an embedding, named the *driving vector*.
The X2Face network architecture is described in Section 3.1, and the self-supervised training framework in Section 3.2. In addition we make two further contributions. First, we propose a method for linearly regressing from a set of labels (e.g. for head pose) or features (e.g. from audio) to the *driving vector*; this is described in Section 4. The performance is evaluated in Section 5, where we show (i) the robustness of the generated results compared to state-of-the-art self-supervised [45] and supervised [1] methods; and (ii) the controllability of the network using other modalities, such as audio or pose. The second contribution, described in Section 6, shows how the *embedded* face representation can be used for video face editing, e.g. adding facial decorations in the manner of [31] using multiple or just a single *source* frame.

2 Related work

Explicit modelling of faces for image generation. Traditionally facial animation (or puppeteering) given one image was performed by fitting a 3DMM and then modifying the estimated parameters [3]. Later work has built on the

fitting of 3DMMs by including high level details [34,41], taking into account additional images [33] or 3D scans [4], or by learning 3DMM parameters directly from RGB data without ground truth labels [39,2]. Please refer to Zollhöfer et. al. [46] for a survey.

Given a driving and source video sequence, a 3DMM or 3D mesh can be obtained and used to model both the driving and source face [43,40,10]. The estimated 3D is used to transform the expression of the source face to match that of the driving face. However, this requires additional steps to transfer the hidden regions (e.g. the teeth). As a result, a neural network conditioned on a single driving image can be used to predict higher level details to fill in these hidden regions [25].

Motivated by the fact that a 3DMM approach is limited by the components of the corresponding morphable model, which may not model the full range of required expressions/deformations and the higher level details, [1] propose a 2D warping method. Given only one source image, [1] use facial landmarks in order to warp the expression of one face onto another. They additionally allow for fine scale details to be transferred by monitoring changes in the driving video.

An interesting related set of works consider how to frontalise a face in a still image using a generic reference face [14], transferring expressions of an actor to an avatar [35] and swapping one face with another [20,24].

Learning based approaches for image generation. There is a wealth of literature on supervised/self-supervised approaches; here we review only the most relevant work. Supervised approaches for controlling a given face learn to model factors of variation (e.g. lighting, pose, etc.) by conditioning the generated image on known ground truth information which may be head pose, expression, or landmarks [44,21,42,5,12,30]. This requires a training dataset with known pose or expression information which may be expensive to obtain or require subjective judgement (e.g. in determining the expression). Consequently, self-supervised and unsupervised approaches attempt to automatically learn the required factors of variation (e.g. optical flow or pose) without labelling. This can be done by maximising mutual information [7] or by training the network to synthesise future video frames [29,11].

Another relevant self-supervised method is CycleGAN [45] which learns to transform images of one domain into those of another. While not explicitly devised for this task, as CycleGAN learns to be cycle-consistent, the transformed images often bear semantic similarities to the original images. For example, a CycleGAN model trained to transform images of one person's face (domain A) into those of another (domain B), will often learn to map the pose/position/expression of the face in domain A onto the generated face from domain B.

Using multi-modal setups to control image generation. Other modalities, such as audio, can control image generation by using a neural network that learns the relationship between audio and correlated parts in corresponding images. Examples are controlling the mouth with speech [8,38], controlling a head with audio and a known emotional state [16], and controlling body movement with music [36].

Our method has the benefits of being self-supervised and the ability to control the generation process from other modalities without requiring explicit modelling of the face. Thus it is applicable to other domains.

3 Method

This section introduces the network architecture in Section 3.1, followed by the curriculum strategy used to train the network in Section 3.2.

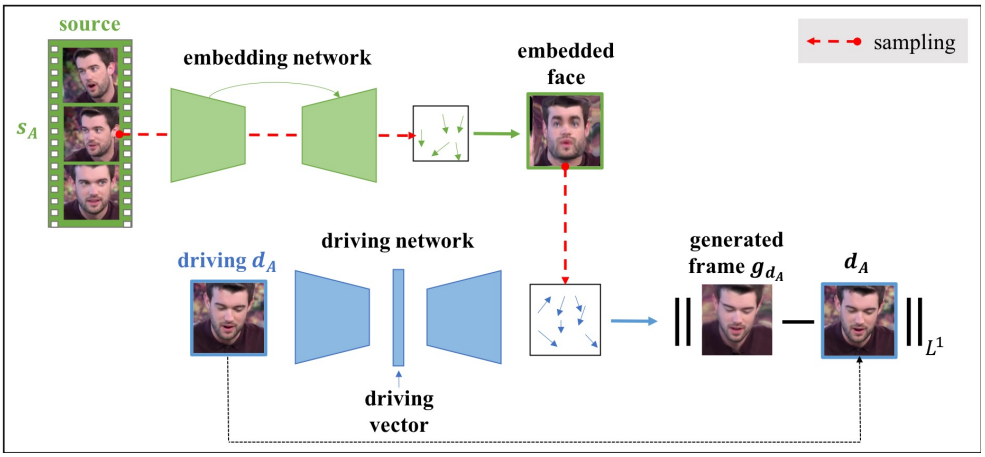


Fig. 2: An overview of X2Face during the initial training stage. Given multiple frames of a video (here 4 frames), one frame is designated the *source* frame and another the *driving* frame. The *source* frame is input to the *embedding network*, which learns a sampler to map pixels from the *source* frame to the *embedded* face. The *driving* frame is input to the *driving network*, which learns to map pixels from the *embedded* face to the *generated* frame. The *generated* frame should have the identity of the *source* frame and the pose/expression of the *driving* frame. In this training stage, as the frames are from the same video, the *generated* and *driving* frames should match. However, at test time the identities of the *source* and *driving* face can differ.

3.1 Architecture

The network takes two inputs: a *driving* and a *source* frame. The *source* frame is input to the *embedding network* and the *driving* frame to the *driving network*. This is illustrated in Fig. 2. Precise architectural details are given in the supplementary material in Section A.1 and Section A.2.

Embedding network. The *embedding network* learns a bilinear sampler to determine how to map from the *source* frame to a face representation, the *embedded* face. The architecture is based on U-Net [32] and pix2pix [15]; the output is a 2-channel image (of the same dimensions as the *source* frame) that encodes the flow $\delta x, \delta y$ for each pixel.

While the *embedding network* is not explicitly forced to frontalise the *source* frame, we observe that it learns to do so for the following reason. Because the *driving network* samples from the *embedded* face to produce the *generated* frame without knowing the pose/expression of the *source* frame, it needs the *embedded* face to have a common representation (e.g. be frontalised) across *source* frames with differing poses and expressions.

Driving network. The *driving network* takes a *driving* frame as input and learns a bilinear sampler to transform pixels from the *embedded* face to produce the *generated* frame. It has an encoder-decoder architecture. In order to sample correctly from the *embedded* face and produce the *generated* frame, the latent embedding (the *driving vector*) must encode pose/expression/zoom/other factors of variation.

3.2 Training the network

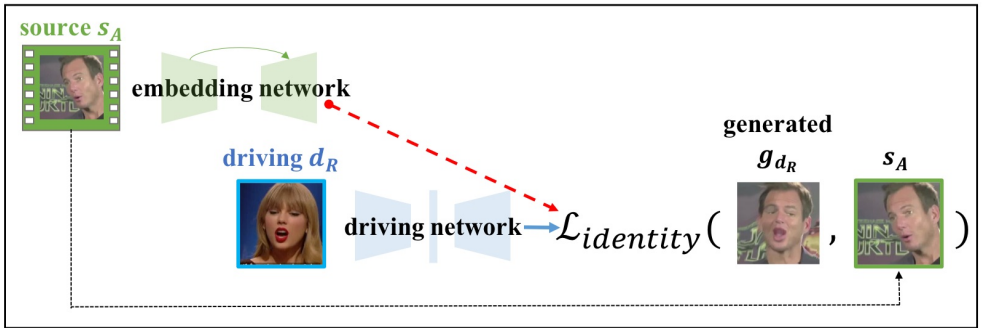


Fig. 3: The identity loss function when the *source* and *driving* frames are of different identities. This loss enforces that the *generated* frame has the same identity as the *source* frame.

The network is trained with a curriculum strategy using two stages. The first training stage (**I**) is fully self-supervised. In the second training stage (**II**), we make use of a CNN pre-trained for face identification to add additional constraints based on the identity of the faces in the *source* and *driving* frames to finetune the model following training stage (**I**).

I. The first stage (illustrated in Fig. 2) uses only a pixelwise $L1$ loss between the *generated* and the *driving* frames. Whilst this is sufficient to train the network such that the *driving* frame encodes expression and pose, we observe that some face shape information is leaked through the *driving vector* (e.g. the *generated* face becomes fatter/longer depending on the face in the *driving* frame). Consequently, we introduce additional loss functions – called identity loss functions – in the second stage.

II. In the second stage, the identity loss functions are applied to enforce that the identity is the same between the *generated* and the *source* frames irrespective of the identity of the *driving* frame. This loss should mitigate against the face shape leakage discussed in stage **I**. In practice, one *source* frame s_A of identity

A, and two *driving* frames d_A,d_R are used as training inputs; d_A is of identity A and d_R a random identity. This gives two generated frames g_{d_A}, g_{d_R} respectively, which should both be of identity A. Two identity loss functions are then imposed: $\mathcal{L}_{\text{identity}}(d_A, g_{d_A})$ and $\mathcal{L}_{\text{identity}}(s_A, g_{d_R})$. $\mathcal{L}_{\text{identity}}$ is implemented using a network pre-trained for identity to measure the similarity of the images in feature space by comparing appropriate layers of the network (i.e. a content loss as in [13,6]). The precise layers are chosen based on whether we are considering g_{d_A} or g_{d_R}:

1. $\mathcal{L}_{identity}(d_A, g_{d_A})$. g_{d_A} should have the same identity, pose and expression as d_A so we use the photometric $L1$ loss and a $L1$ content loss on the Conv2-5 and Conv7 layers (i.e. layers that encode both lower/higher level information such as pose/identity) between g_{d_A} and d_A.
2. $\mathcal{L}_{identity}(s_A, g_{d_R})$ *(Fig. 3)*. g_{d_R} should have the identity of s_A but the pose and expression of d_R. Consequently, we cannot use the photometric loss but only a content loss. We minimise a $L1$ content loss on the Conv6-7 layers (i.e. layers encoding higher level identity information) between g_{d_A} and s_A.

The pre-trained network used for these losses is the 11-layer VGG network (configuration A) [37] trained on the VGG-Face Dataset [26].

4 Controlling the image generation with other modalities

Given a trained X2Face network, the *driving vector* can be used to control the *source* face with other modalities such as audio or pose.

4.1 Pose

Instead of controlling the generation with a *driving* frame, we can control the head pose of the *source* face using a pose code such that when varying the code's pitch/yaw/roll angles, the *generated* frame varies accordingly. This is done by learning a forward mapping $f_{p\to v}$ from head pose p to the *driving vector* v such that $f_{p\to v}(p)$ can serve as a modified input to the *driving network*'s decoder. However, this is an ill-posed problem; directly using this mapping loses information, as the *driving vector* encodes more than just pose.
As a result, we use vector arithmetic. Effectively we drive a *source* frame with itself but modify the corresponding *driving vector* v_{emb}^{source} to remove the pose of the *source* frame p_{source} and incorporate the new driving pose $p_{driving}$. This gives:

$$v_{emb}^{driving} = v_{emb}^{source} + v_{emb}^{\Delta pose} = v_{emb}^{source} + f_{p\to v}(p_{driving} - p_{source}). \quad (1)$$

However, VoxCeleb [23] does not contain ground truth head pose, so an additional mapping $f_{v\to p}$ is needed to determine $p_{source} = f_{v\to p}(v_{emb}^{source})$.

$f_{v\to p}$. $f_{v\to p}$ is trained to regress p from v. It is implemented using a fully connected layer with bias and trained using an L1 loss. Training pairs (v, p) are obtained using an annotated dataset with image to pose labels p; v is obtained by passing the image through the encoder of the *driving network*.

$f_{p\to v}$. $f_{p\to v}$ is trained to regress v from p. It is implemented using a fully-connected linear layer with bias followed by batch-norm. When $f_{v\to p}$ is known, this function can be learnt directly on VoxCeleb by passing an image through X2Face to get the *driving vector* v and $f_{v\to p}(v)$ gives the pose p.

4.2 Audio

Audio data from the videos in the VoxCeleb dataset can be used to drive a *source* face in a manner similar to that of pose by driving the *source* frame with itself but modifying the *driving vector* using the audio from another frame. The forward mapping $f_{a\to v}$ from audio features a to the corresponding *driving vector* v is trained using pairs of audio features a and driving vectors v. These can be directly extracted from VoxCeleb (so no backward mapping $f_{v\to a}$ is required). a is obtained by extracting the 256D audio features from the neural network in [9] and the 128D v by passing the corresponding frame through the *driving network*'s encoder. Ordinary least squares linear regression is then used to learn $f_{a\to v}$ after first normalising the audio features to $\sim N(0,1)$. No normalisation is used when employing the mapping to drive the frame generation; this amplifies the signal, visually improving the generated results.
As learning the function $f_{a\to v} : \mathbb{R}^{1\times 256} \to \mathbb{R}^{1\times 128}$ is under-constrained, the embedding learns to encode some pose information. Therefore, we additionally use the mappings $f_{p\to v}$ and $f_{v\to p}$ described in Section 4.1 to remove this information. Given driving audio features $a_{driving}$ and the corresponding, non-modified *driving vector* v_{emb}^{source}, the new *driving vector* $v_{emb}^{driving}$ is then

$$v_{emb}^{driving} = v_{emb}^{source} + f_{a\to v}(a_{driving}) - f_{a\to v}(a_{source}) - f_{p\to v}(p_{audio} - p_{source}),$$

where $p_{source} = f_{v\to p}(v_{emb}^{source})$ is the head pose of the frame input to the *driving network* (i.e. the *source* frame), $p_{audio} = f_{v\to p}(f_{a\to v}(a_{driving}))$ is the pose information contained in $f_{a\to v}(a_{driving})$, and a_{source} is the audio feature vector corresponding to the *source* frame.

5 Experiments

This section evaluates X2Face by first performing an ablation study in Section 5.1 on the architecture and losses used for training, followed by results for controlling a face with a *driving* frame in Section 5.2, pose information in Section 5.3, and audio information in Section 5.4.
Training. X2Face is trained on the VoxCeleb video dataset [23] using dlib [18] to crop the faces to 256×256. The identities are randomly split into train/val/test identities (with a split of 75/15/10) and frames extracted at one fps to give 900,764 frames for training and 125,131 frames for testing.
The model is trained in PyTorch [27] using SGD with momentum 0.9 and batch-size of 16. First, it is trained just with $L1$ loss, and a learning rate of 0.001. The learning rate is decreased by a factor of 10 when the loss plateaus. Once the loss

converges, the identity losses are incorporated and are weighted as follows: (i) for same identities to be as strong as the photometric $L1$ loss at each layer; (ii) for different identities to be 1/10 the size of the photometric loss at each layer. This training phase is started with a learning rate of 0.0001.

Testing. The model can be tested using either a single or multiple *source* frames. The reasoning for this is that if the *embedded* face is stable (e.g. different facial regions always map to the same place on the *embedded* face), we expect to be able to combine multiple *source* frames by averaging over the *embedded* faces.

5.1 Architecture studies

To quantify the utility of using additional views at *test* time and the benefit of the curriculum strategy for training the network (i.e. using the identity losses explained in Section 3.2), we evaluate the results for these different settings on a left-out test set of VoxCeleb. We consider 120K *source* and *driving* pairs where the *driving* frame is from the same video as the *source* frames; thus, the *generated* frame should be the same as the *driving* frame. The results are given in Table 1.

Table 1: $L1$ reconstruction error on the test set, comparing the *generated* frame to the ground truth frame (in this case the *driving* frame) for different training/testing setups. Lower is better for $L1$ error. Additionally, we give the percentage improvement over the $L1$ error for the model trained with only training stage **I** and tested with a single *source* frame. In this case, higher is better

Training strategy	**# of *source* frames at test time**	$L1$ **error**	**% Improvement**
Training stage **I**	1	0.0632	0%
Training stage **II**	1	0.0630	0.32%
Training stage **I**	3	0.0524	17.14%
Training stage **II**	3	0.0521	17.62%

The results in Table 1 confirm that both training with the curriculum strategy and using additional views at *test* time improve the reconstructed image. Section A.3 in the supplementary material includes qualitative results and shows that using additional *source* frames when testing is especially useful if a face is seen at an extreme pose in the initial *source* frame.

5.2 Controlling image generation with a *driving* frame

The motivation of our architecture is to be able to map the expression and pose of a *driving* frame onto a *source* frame *without* any annotations on expression or pose. This section demonstrates that X2Face does indeed achieve this, as a set of *source* frames can be controlled with a driving video and generate realistic results. We compare to two methods: CycleGAN [45] which uses *no* labels and [1] which is designed top down and demonstrates impressive results. Additional

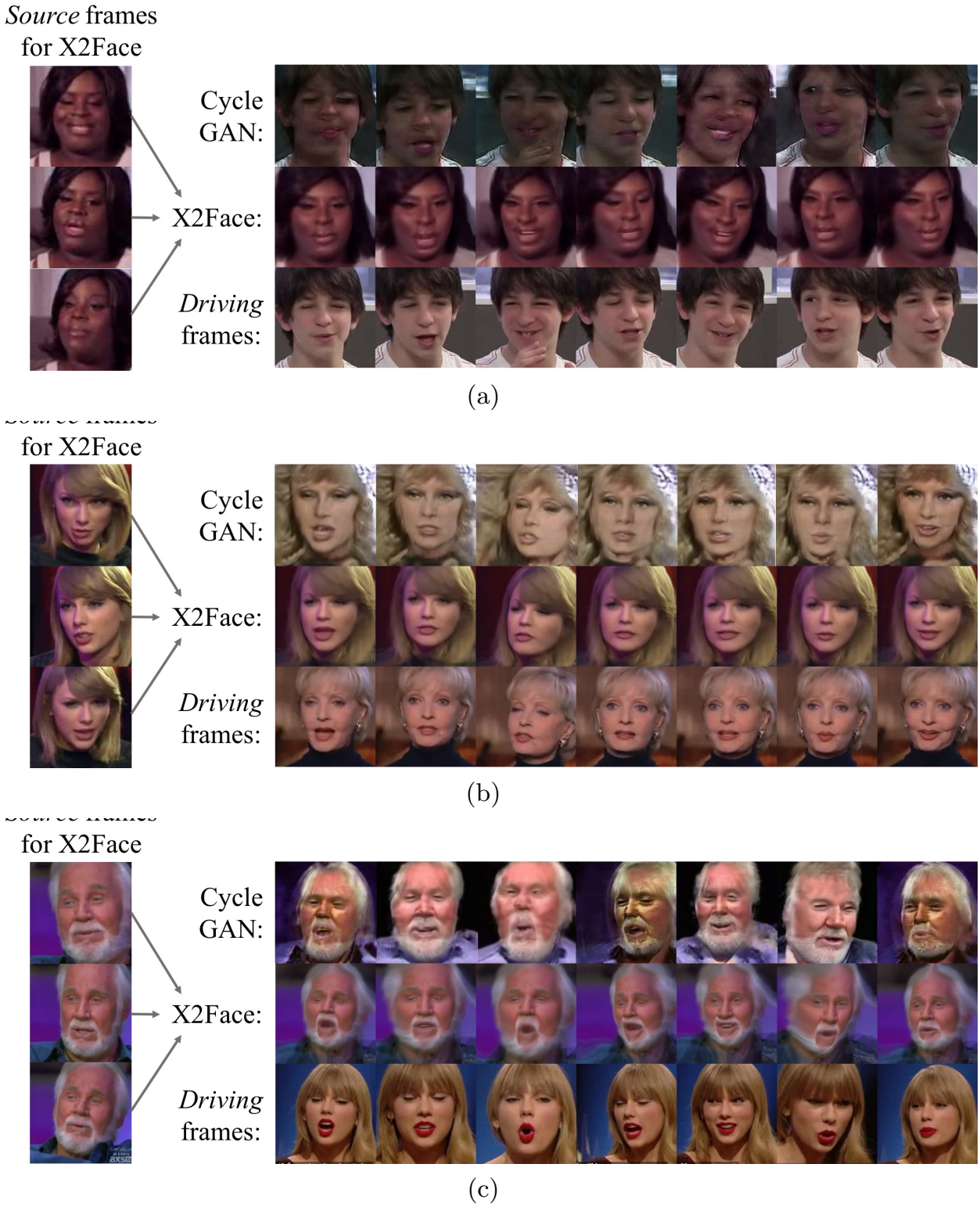


Fig. 4: Comparison of X2Face's *generated* frames to those of CycleGAN given a driving video sequence. Each example shows from bottom to top: the *driving* frame, our *generated* result and CycleGAN's generated result. To the left, *source* frames for X2Face are shown (at test time CycleGAN does not require *source* frames, as it is has been trained to map between the given *source* and *driving* identities). These examples demonstrate multiple benefits of our method. *First*, X2Face is capable of preserving the face shape of the source identity (top row) whilst driving the pose and expression according to the *driving* frame (bottom row); CycleGAN correctly keeps pose and expression but loses information about face shape and geometry when given too few training images as in example (a) (whereas X2Face requires no training samples for new identities). *Second*, X2Face has temporal consistency. CycleGAN samples from the latent space, so

qualitative results are given in Fig. 12 in the supplementary material and in the accompanying video[1].

Comparison to CycleGAN [45]. CycleGAN learns a mapping from a given domain (in this case a given identity A) to another domain (in this case another identity B). To compare to their method for a given pair of identities, we take all images of the given identities (so images may come from different video tracks) to form two sets of images: one set corresponding to identity A and the other to B. We then train their model using these sets. To compare, for a given *driving* frame of identity A, we visualise their *generated* frame from identity B which is compared to that of X2Face.
The results in Fig. 4 illustrate multiple benefits. First, X2Face generalises to unseen pairs of identities at test time given only a *source* and *driving* frame. CycleGAN is trained on pairs of identities, so if there are too few example images, it fails to correctly model the shape and geometry of the *source* face, producing unrealistic results. Additionally, our results have better temporal coherence (i.e. consistent background/hair style/etc. across *generated* frames), as X2Face transforms a given frame whereas CycleGAN samples from a latent space.

Comparison to Averbuch-Elor et. al. [1]. We compare to [1] in Fig. 5. There are two significant advantages of our formulation over theirs: first, we can handle more significant pose changes in the driving video and *source* frame (Fig. 5b-c). Second, ours has fewer assumptions: (1)[1] assumes that the first frame of the driving video is in a frontal pose with a neutral expression and that the *source* frame also has a neutral expression (Fig. 5d). (2) X2Face can be used when given a single *driving* frame whereas their method requires a video so that the face can be tracked and the tracking used to expand the number of correspondences and to obtain high level details.
While this is not the focus of this paper, our method can be augmented with the ideas from these methods. For example, as inspired by [1], we can perform simple post-processing to add higher level details (Fig. 5a, X2Face+p.p.) by transferring hidden regions using Poisson editing [28].

5.3 Controlling the image generation with pose

Before reporting results on controlling the *driving vector* using pose, we validate our claim that the *driving vector* does indeed learn about pose. To do this, we evaluate how accurately we can predict the three head pose angles – yaw, pitch and roll – given the 128D *driving vector*.

Pose predictor. To train the pose predictor which also serves as $f_{v\to p}$ (Section 4.1), the 25,993 images in the AFLW dataset [19] are split into train/val set, leaving out the 1,000 test images from [22] as test set. The results on the test set are reported in Table 2 confirming that the *driving vector* learns about head

[1] http://www.robots.ox.ac.uk/~vgg/research/unsup_learn_watch_faces/x2face.html

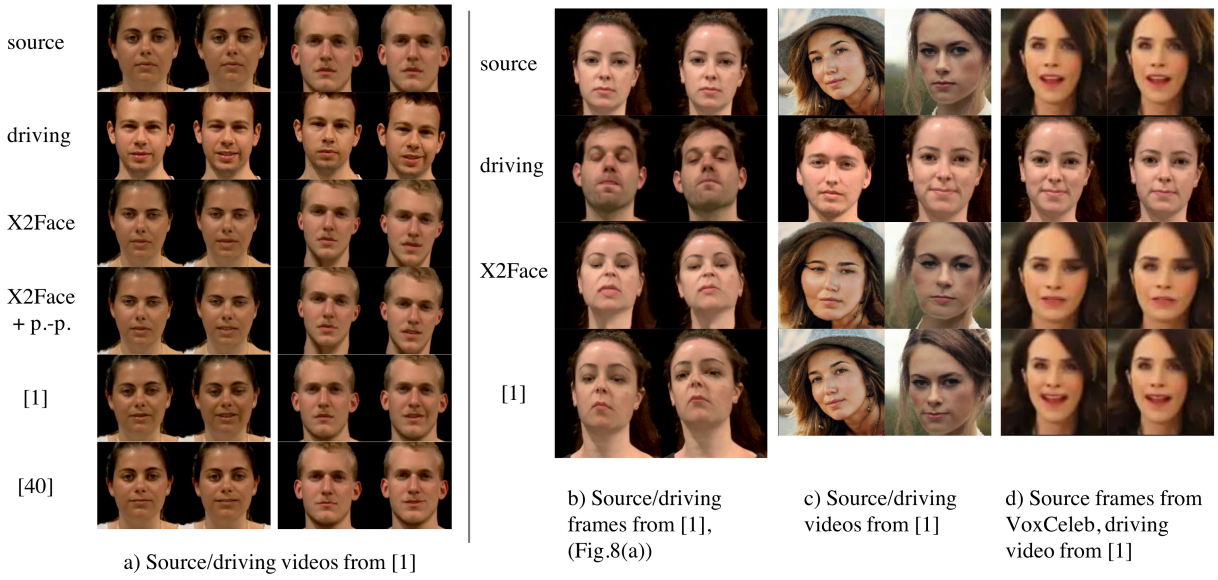


Fig. 5: Comparison of X2Face to supervised methods. In comparison to [1]: X2Face matches (b) pitch, and (c) roll and yaw; and X2Face can handle non-neutral expressions in the *source* frame (d). As with other methods, post-processing (X2Face + p.-p.) can be applied to add higher level details (a).

pose without having been trained on pose labels, as the results are comparable to those of a network directly trained for this task.
We then use $f_{v\to p}$ to train $f_{p\to v}$ (Section 4.1) and present generated frames for different, unseen test identities using the learnt mappings in Fig. 6. The *source* frame corresponds to p_{source} in Section 4.1 while $p_{driving}$ is used to vary one head pose angle while keeping the others fixed.

Table 2: MAE in degrees using the *driving vector* for head pose regression (lower is better). Note that the linear pose predictor from the *driving vector* performs only slightly worse than a supervised method [22], which has been trained for this task

Method	**Roll**	**Pitch**	**Yaw**	**MAE**
X2Face	5.85	7.59	14.62	9.36
KEPLER [22] (supervised)	8.75	5.85	6.45	7.02

5.4 Controlling the image generation with audio input

This section presents qualitative results for using audio data from videos in the VoxCeleb dataset to drive the *source* frames. The VoxCeleb dataset consists of videos of interviews, suggesting that the audio should be especially correlated with the movements of the mouth. [9]'s model, trained on the BBC-Oxford 'Lip

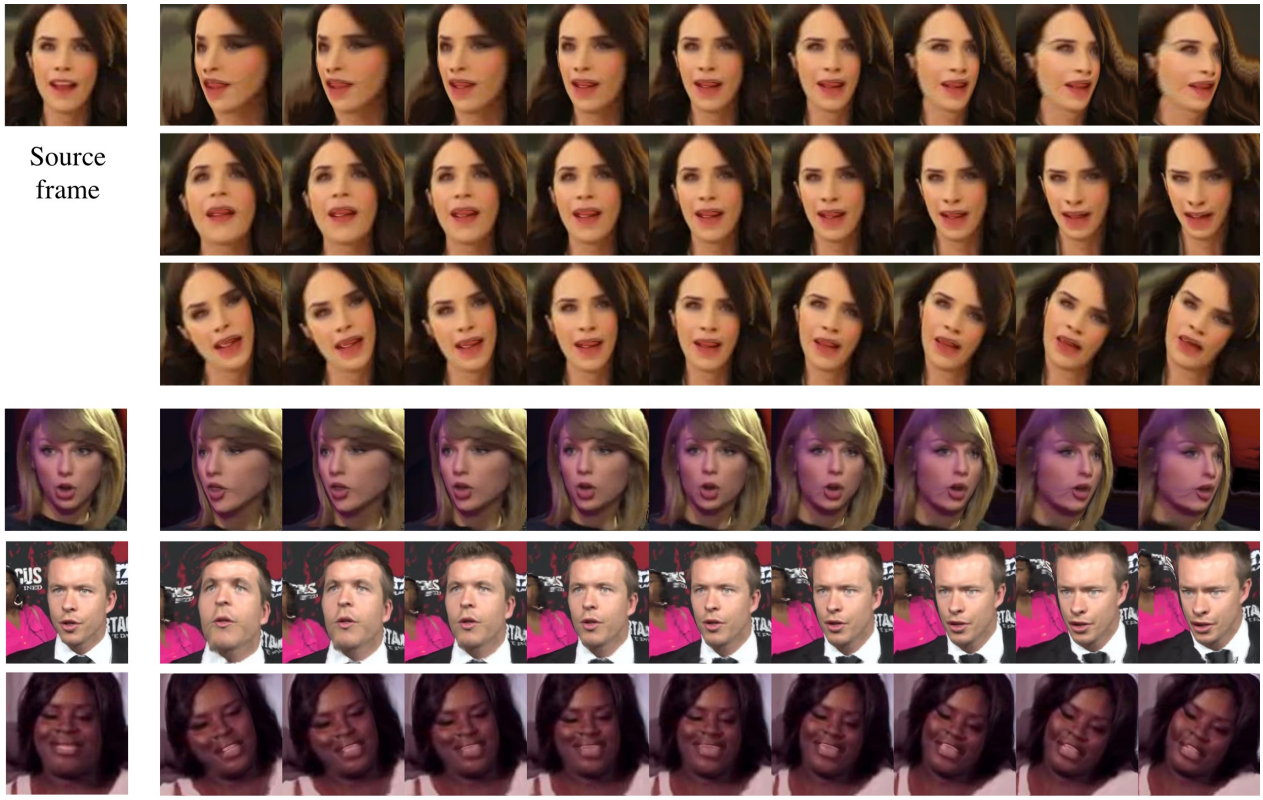


Fig. 6: Controlling image generation with pose code vectors. Results are shown for a single *source* frame which is controlled using each of the three head pose angles for the same identity (top three rows) and for different identities (bottom three rows). For further results and a video animation, we refer to Fig. 15 in the supplementary material and to the accompanying video. Whilst some artefacts are visible, the method allows the head pose angles to be controlled separately.

Reading in the Wild' dataset (LRW), is used to extract audio features. We use the 256D vector activations of the last fully connected layer of the audio stream (FC7) for a 0.2s audio signal centred on the *driving* frame (the frame occurs half way through the 0.2s audio signal).

A potential source of error is the domain gap between the LRW dataset and VoxCeleb, as [9]'s model is not fine-tuned on the VoxCeleb dataset which contains much more background noise than the LRW dataset. Thus, their model has not necessarily learnt to become indifferent to this noise. However, our model is relatively robust to this problem; we observe that the mouth movements in the *generated* frames are reasonably close to what we would expect from the sounds of the corresponding audio, as demonstrated in Fig. 7. This is true even if the person in the video is not speaking and instead the audio is coming from an interviewer. However, there is some jitter in the generation.

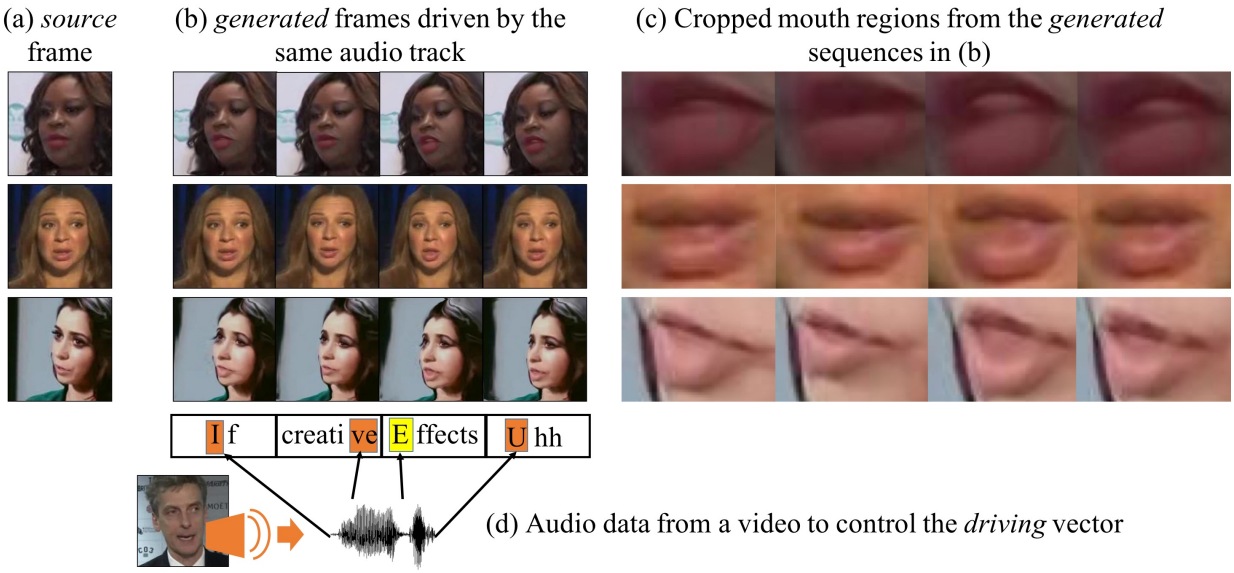


Fig. 7: Controlling image generation with audio information. We show how the same sounds affect various *source* frames; if our model is working well then the *generated* mouths should behave similarly. (a) shows the *source* frames. (b) shows the *generated* frames for a given audio sound which is visualised in (d) by the coloured portion of the word being spoken. As most of the change is expected to be in the mouth region, the cropped mouth regions are additionally visualised in (c). The audio comes from a native British speaker. As can be seen, in all generated frames, the mouths are more closed at the "ve" and "I" and more open at the "E" and "U". Another interesting point is that for the "Effects" frame, the audio is actually coming from an interviewer, so while the frame corresponding to the audio has a closed mouth, the *generated* results still open the mouth.

6 Using the embedded face for video editing

We consider how the *embedded* face can be used for video editing. This idea is inspired by the concept of an unwrapped mosaic [31]. We expect the *embedded* face to be pose and expression invariant, as can be seen qualitatively across the example *embedded* faces shown in the paper. Therefore, the *embedded* face can be considered as a UV texture map of the face and drawn on directly.

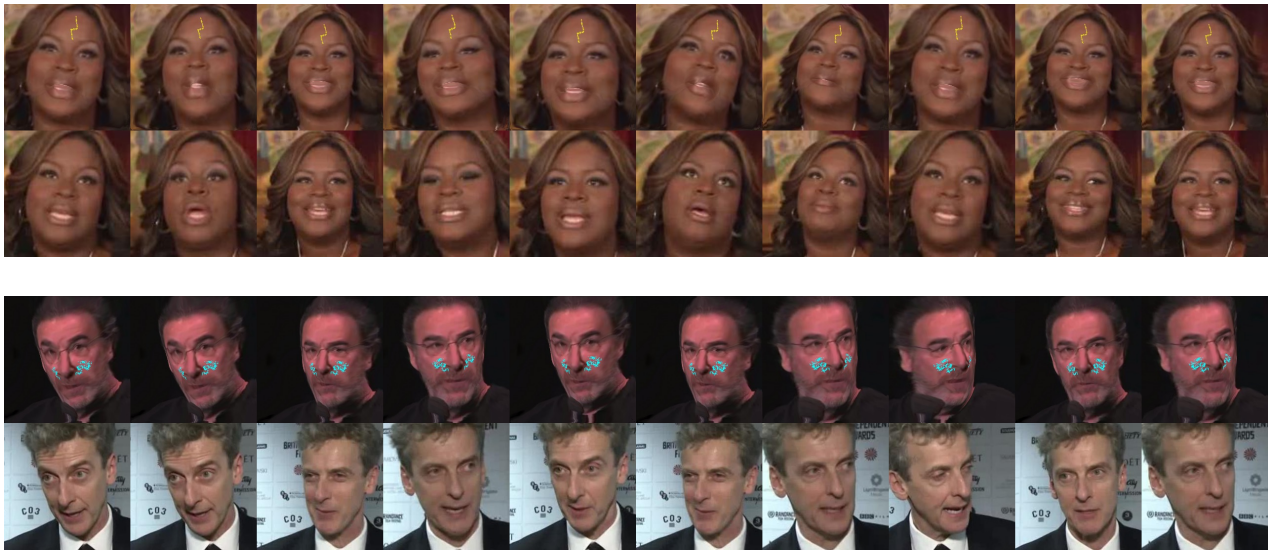
This task is executed as follows. A *source* frame (or set of *source* frames) is extracted and input to the *embedding network* to obtain the *embedded* face. The *embedded* face can then be drawn on using an image or other interactive tool. A video is reconstructed using the modified *embedded* face which is driven by a set of *driving* frames. Because the *embedded* face is stable across different identities, a given edit can be applied across different identities. Example edits are shown in Fig. 8 and in Fig. 13 in the supplementary material.

7 Conclusion

We have presented a self-supervised framework X2Face for driving face generation using another face. This framework makes no assumptions about the



(a) *Source* frames are input to extract the *embedded* face which is drawn on. The modified *embedded* face is used to *generate* the frames below.



(b) An example sequence of *generated* frames (top row) from the modified *embedded* face controlled using a sequence of *driving* frames (bottom row).

Fig. 8: Example results of the video editing application. (a) For given *source* frames, the *embedded* face is extracted and modified. (b) The modified *embedded* face is used for a sequence of *driving* frames (bottom) and the result is shown (top). Note how for the second example, the blue tattoo disappears behind the nose when the person is seen in profile and how, as above, the modified *embedded* face can be driven using the same or another identity's pose and expression. Best seen in colour. Zoom in for details. Additional examples using the blue tattoo and Harry Potter scar are given in the accompanying video and in Fig. 13 in the supplementary material.

pose, expression, or identity of the input images, so it is more robust to unconstrained settings (e.g. an unseen identity). The framework can also be used with minimal alteration *post* training to drive a face using audio or head pose information. Finally, the trained model can be used as a video editing tool. Our model has achieved all this without requiring annotations for head pose/facial landmarks/depth data. Instead, it is trained self-supervised on a large collection of videos and learns itself to model the different factors of variation.

While our method is robust, versatile, and allows for generation to be conditioned on other modalities, the generation quality is not as high as approaches specifically designed for transforming faces (e.g. [1,17,40]). This opens an interesting avenue of research: how can the approach be modified such that the versatility, robustness, and self-supervision aspects are retained but with the generation quality of these methods that are specifically designed for faces. Finally, as no assumptions have been made that the videos are of faces, it is interesting to consider applying our approach to other domains.

Acknowledgements The authors are grateful to Hadar Averbuch-Elor for helpfully running their model on our data and to Vicky Kalogeiton for suggestions/comments. This work was funded by an EPSRC studentship and EPSRC Programme Grant Seebibyte EP/M013774/1.

References

1. Averbuch-Elor, H., Cohen-Or, D., Kopf, J., Cohen, M.F.: Bringing portraits to life. ACM Transactions on Graphics (Proceeding of SIGGRAPH Asia 2017) (2017)
2. Bas, A., Smith, W.A.P., Awais, M., Kittler, J.: 3D morphable models as spatial transformer networks. In: Proc. ICCV Workshop on Geometry Meets Deep Learning (2017)
3. Blanz, V., Vetter, T.: A morphable model for the synthesis of 3d faces. In: Proc. ACM SIGGRAPH (1999)
4. Booth, J., Roussos, A., Ponniah, A., Dunaway, D., Zafeiriou, S.: Large scale 3d morphable models. IJCV **126**(2-4), 233–254 (Apr 2018)
5. Cao, J., Hu, Y., Yu, B., He, R., Sun, Z.: Load balanced gans for multi-view face image synthesis. arXiv preprint arXiv:1802.07447 (2018)
6. Chen, Q., Koltun, V.: Photographic image synthesis with cascaded refinement networks. In: Proc. ICCV (2017)
7. Chen, X., Duan, Y., Houthooft, R., Schulman, J., Sutskever, I., Abbeel, P.: Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In: NIPS (2016)
8. Chung, J.S., Senior, A., Vinyals, O., Zisserman, A.: Lip reading sentences in the wild. In: Proc. CVPR (2017)
9. Chung, J.S., Zisserman, A.: Out of time: automated lip sync in the wild. In: Workshop on Multi-view Lip-reading, ACCV (2016)
10. Dale, K., Sunkavalli, K., Johnson, M.K., Vlasic, D., Matusik, W., Pfister, H.: Video face replacement. ACM Transactions on Graphics (TOG) (2011)
11. Denton, E.L., Birodkar, V.: Unsupervised learning of disentangled representations from video. In: NIPS (2017)
12. Ding, H., Sricharan, K., Chellappa, R.: Exprgan: Facial expression editing with controllable expression intensity. In: Proc. AAAI (2018)
13. Gatys, L.A., Ecker, A.S., Bethge, M.: Image style transfer using convolutional neural networks. In: Proc. CVPR (2016)
14. Hassner, T., Harel, S., Paz, E., Enbar, R.: Effective face frontalization in unconstrained images. In: Proc. CVPR (2015)
15. Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A.: Image-to-image translation with conditional adversarial networks. In: Proc. CVPR (2017)

16. Karras, T., Aila, T., Laine, S., Herva, A., Lehtinen, J.: Audio-driven facial animation by joint end-to-end learning of pose and emotion. ACM Transactions on Graphics (TOG) (2017)
17. Kim, H., Garrido, P., Tewari, A., Xu, W., Thies, J., Nießner, M., Pérez, P., Richardt, C., Zollhöfer, M., Theobalt, C.: Deep video portraits. Proc. ACM SIGGRAPH (2018)
18. King, D.E.: Dlib-ml: A machine learning toolkit. The Journal of Machine Learning Research **10**, 1755–1758 (2009)
19. Koestinger, M., Wohlhart, P., Roth, P.M., Bischof, H.: Annotated Facial Landmarks in the Wild: A Large-scale, Real-world Database for Facial Landmark Localization. In: Proc. First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies (2011)
20. Korshunova, I., Shi, W., Dambre, J., Theis, L.: Fast face-swap using convolutional neural networks. In: Proc. ICCV (2017)
21. Kulkarni, T.D., Whitney, W.F., Kohli, P., Tenenbaum, J.: Deep convolutional inverse graphics network. In: NIPS (2015)
22. Kumar, A., Alavi, A., Chellappa, R.: KEPLER: keypoint and pose estimation of unconstrained faces by learning efficient H-CNN regressors. In: Proc. Int. Conf. Autom. Face and Gesture Recog. (2017)
23. Nagrani, A., Chung, J.S., Zisserman, A.: VoxCeleb: a large-scale speaker identification dataset. In: INTERSPEECH (2017)
24. Nirkin, Y., Masi, I., Tran, A.T., Hassner, T., Medioni, G.: On face segmentation, face swapping, and face perception. In: Proc. Int. Conf. Autom. Face and Gesture Recog. (2018)
25. Olszewski, K., Li, Z., Yang, C., Zhou, Y., Yu, R., Huang, Z., Xiang, S., Saito, S., Kohli, P., Li, H.: Realistic dynamic facial textures from a single image using gans. In: Proc. ICCV (2017)
26. Parkhi, O.M., Vedaldi, A., Zisserman, A.: Deep face recognition. In: Proc. BMVC. (2015)
27. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in PyTorch (2017)
28. Pérez, P., Gangnet, M., Blake, A.: Poisson image editing. ACM Transactions on graphics (TOG) (2003)
29. Pătrăucean, V., Handa, A., Cipolla, R.: Spatio-temporal video autoencoder with differentiable memory. In: NIPS (2016)
30. Qiao, F., Yao, N., Jiao, Z., Li, Z., Chen, H., Wang, H.: Geometry-contrastive generative adversarial network for facial expression synthesis. arXiv preprint arXiv:1802.01822 (2018)
31. Rav-Acha, A., Kohli, P., Rother, C., Fitzgibbon, A.: Unwrap mosaics: A new representation for video editing. In: ACM Transactions on Graphics (TOG) (2008)
32. Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional networks for biomedical image segmentation. In: Proc. MICCAI (2015)
33. Roth, J., Tong, Y., Liu, X.: Adaptive 3D face reconstruction from unconstrained photo collections. In: Proc. CVPR (2016)
34. Saito, S., Wei, L., Hu, L., Nagano, K., Li, H.: Photorealistic facial texture inference using deep neural networks. In: Proc. CVPR (2017)
35. Saragih, J.M., Lucey, S., Cohn, J.F.: Real-time avatar animation from a single image. In: Proc. Int. Conf. Autom. Face and Gesture Recog. (2011)
36. Shlizerman, E., Dery, L., Schoen, H., Kemelmacher-Shlizerman, I.: Audio to body dynamics. Proc. CVPR (2018)

37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations (2015)
38. Suwajanakorn, S., Seitz, S.M., Kemelmacher-Shlizerman, I.: Synthesizing Obama: learning lip sync from audio. ACM Transactions on Graphics (TOG) (2017)
39. Tewari, A., Zollhöfer, M., Kim, H., Garrido, P., Bernard, F., Perez, P., Theobalt, C.: Mofa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In: Proc. ICCV (2017)
40. Thies, J., Zollhöfer, M., Stamminger, M., Theobalt, C., Nießner, M.: Face2Face: Real-time Face Capture and Reenactment of RGB Videos. In: Proc. CVPR (2016)
41. Tran, A.T., Hassner, T., Masi, I., Paz, E., Nirkin, Y., Medioni, G.: Extreme 3D face reconstruction: Seeing through occlusions. In: Proc. CVPR (2018)
42. Tran, L., Yin, X., Liu, X.: Disentangled representation learning gan for pose-invariant face recognition. In: Proc. CVPR (2017)
43. Vlasic, D., Brand, M., Pfister, H., Popović, J.: Face transfer with multilinear models. ACM Transactions on Graphics (TOG) (2005)
44. Worrall, D.E., Garbin, S.J., Turmukhambetov, D., Brostow, G.J.: Interpretable transformations with encoder-decoder networks. In: Proc. ICCV (2017)
45. Zhu, J.Y., Park, T., Isola, P., Efros, A.A.: Unpaired image-to-image translation using cycle-consistent adversarial networks. Proc. ICCV (2017)
46. Zollhöfer, M., Thies, J., Garrido, P., Bradley, D., Beeler, T., Pérez, P., Stamminger, M., Nießner, M., Theobalt, C.: State of the art on monocular 3D face reconstruction, tracking, and applications. In: Proc. Eurographics (2018)

Supplementary Material

We provide additional details about X2Face's architecture and the training and testing setup in Section A, and more qualitative results in Section B.

A Additional details on architectures and training.

Section A.3 provides qualitative results on using additional views at *test* time and the curriculum strategy. Section A.1 and Section A.2 give additional details about X2Face's two subnetworks: the *embedding network* and the *driving network*.

A.1 Embedding network

The *embedding network* is based on the pix2pix [15] version of the U-Net architecture [32], except that the last layer is changed to a 2-channel layer. The resulting $2 \times 256 \times 256$ gives the sampler which determines how to sample from the *source* frame to obtain the *embedded* face. The exact architecture is given in Fig. 9.

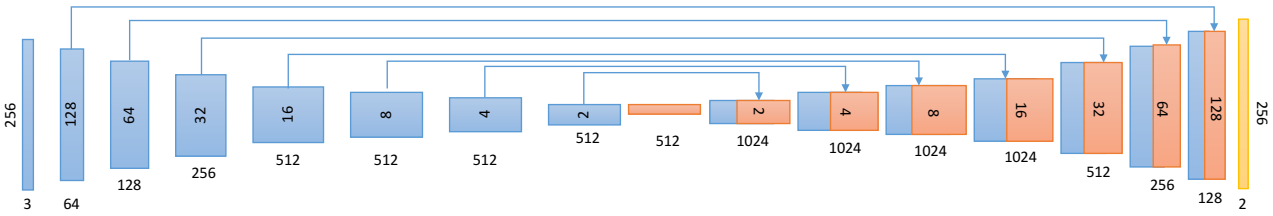


Fig. 9: *Embedding Network*: In the encoder, each convolution is followed by a leaky ReLU with factor 0.2 and then a batch-norm layer (except for the first which has no batch-norm). The convolutional filter sizes are 4×4 and the stride/padding is 2/1. In this decoder, the layers from the encoder are concatenated using skip connections, doubling the channel size. In the decoder, the sequence of executions following each convolutional layer is: ReLU, bilinear upsampling, batch-norm, concatenation (with the corresponding encoder layer). The last layer (yellow) does not have a concatenation. The final $2 \times 256 \times 256$ result is passed through a tanh layer to give X2Face's prediction for how to sample from the *source* frames. The convolutional filter sizes in the decoder are 3×3 and the stride/padding is 1/1.

A.2 Driving network

The *driving network* is an encoder-decoder network that uses the same layer sizes as the *embedding network* but does not have any skip connections. The *driving vector* corresponds to the vector resulting from the encoder portion of the network, which is a 128D vector. The architecture is shown in Fig. 10.

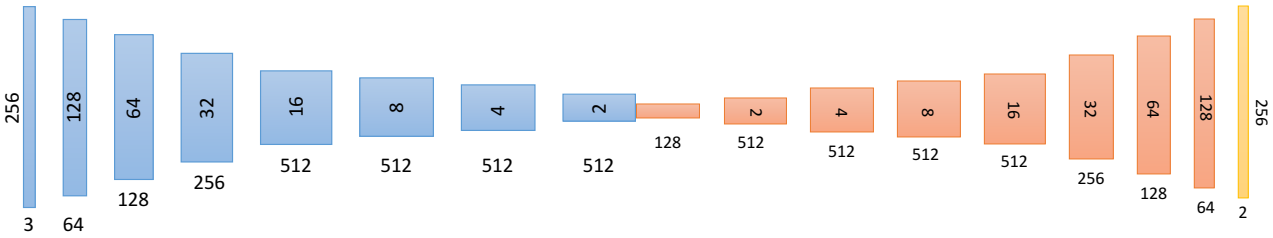


Fig. 10: *Driving Network*: The encoder is similar to the encoder of the *embedding network*; each convolution is followed by a leaky ReLU (factor 0.2) and then a batch-norm layer (except for the first which has no batch-norm); the convolutional filter sizes are 4×4 and the stride/padding is 2/1. In this decoder, there are no skip connections. The *driving vector* has channel size 128. In the decoder, the sequence of executions following each convolutional layer is: ReLU, bilinear upsampling, batch-norm. The final $2 \times 256 \times 256$ result is passed through a tanh layer to give X2Face's prediction for how to sample from the *embedded* face. The convolutional filter sizes in the decoder are 3×3 and the stride/padding is 1/1.

A.3 Qualitative results for different training/testing setups

We consider the following 4 different training/testing settings:

1. **Training stage I, Single source (S):** Training with photometric $L1$ loss only and testing with a single *source* frame.
2. **Training stage I, Multi-source (M):** Training with photometric $L1$ loss only and testing with multiple *source* frames.
3. **Training stage II, S:** Training with photometric $L1$ loss and identity losses as described in Section 3.2 in the paper and testing with a single *source* frame.
4. **Training stage II, M:** Training with photometric $L1$ loss and identity losses as described in Section 3.2 in the paper and testing with multiple *source* frames.

As can be seen in Fig. 11, testing with multiple *source* frames makes our method more robust, as the *generated* frame is not so unstable with respect to the choice of the *source* frame. This improves the quality and sharpness of the *generated* frames.
Although the results get better quantitatively when adding the identity loss functions (as discussed in the paper), we observe that our method produces visually very similar and convincing results when trained with the photometric $L1$ loss only. The use of the identity content losses is the only point where we make use of some form of labelled supervision, as we use a pre-trained network that was trained for identity classification. Using only the photometric loss results in a *completely self-supervised* framework.

B Additional qualitative results

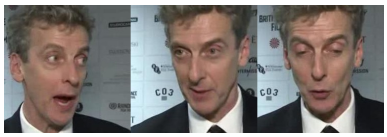
We present more qualitative results comparing X2Face to CycleGAN [45] in Fig. 12 and show additional results for controlling the head pose of a *source*



(a) Single *source* frame



(b) *Embedded face* from a single *source* frame



(c) Multiple *source* frames



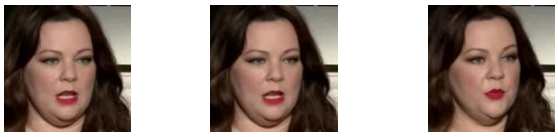
(d) *Embedded* face from multiple *source* frames



(e) Frame sequences *generated* for the same *driving* frames with different loss function/architecture choices.

Fig. 11: A comparison of different training/testing settings. (a) The *source* frame used when testing using a single *source* frame. (b) The *embedded* face for this single *source* frame. (c) The *source* frames used when testing with multiple *source* frames. (d) The *embedded* face from these multiple *source* frames. (e) *Generated* frames for different training stages and testing strategies. As the *driving* frame is from the same video, the *driving* frame corresponds to the ground truth and is shown in the bottom row. When using a single *source* frame for testing, the model is sensitive to the quality/pose of the *source* frame (see rows for *Training stage I, Single source (S)* and *Training stage II, S*). Using additional *source* frames and averaging over the resulting *embedded* face produces a better *embedded* face representation and higher quality results (see rows for *Training stage I, Multi-source (M)* and *Training stage II, M*).

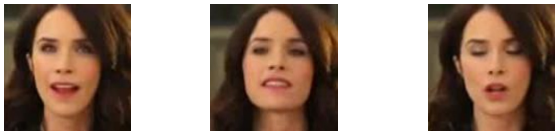
frame using a pose vector in Fig. 15. Finally, additional qualitative results for using X2Face for video editing are given in Fig. 13 and Fig. 14.
For further demonstrations of X2Face in action, we refer to the accompanying video which can be found at http://www.robots.ox.ac.uk/~vgg/research/unsup_learn_watch_faces/x2face.html.



(a) *source* frames



(b)

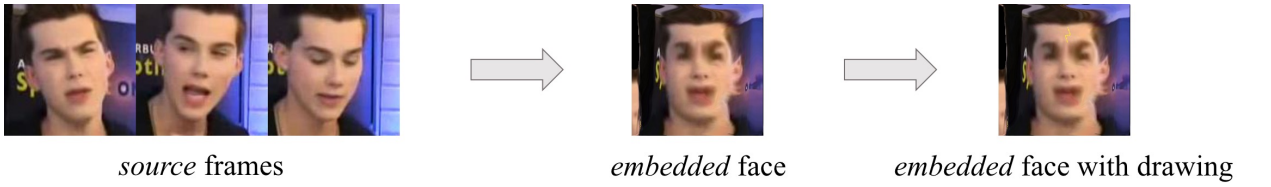


(c) *source* frames



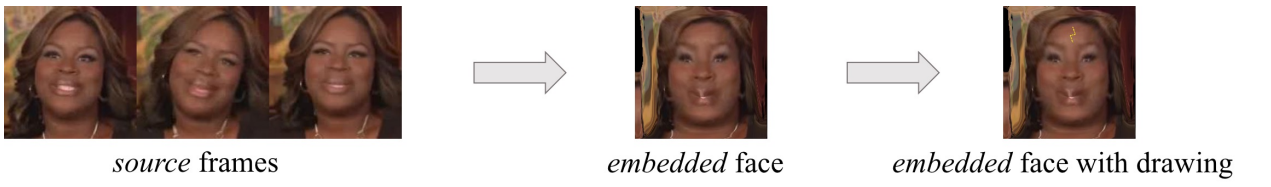
(d)

Fig. 12: More qualitative results comparing X2Face to CycleGAN. Note how X2Face preserves the hair/background setting from the *source* frames, resulting in temporal coherence across the *generated* frames.





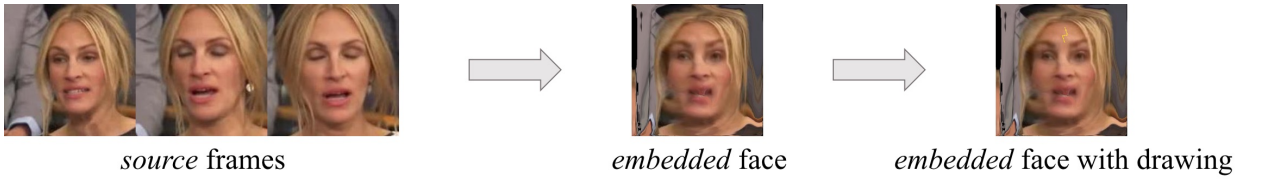
(b)





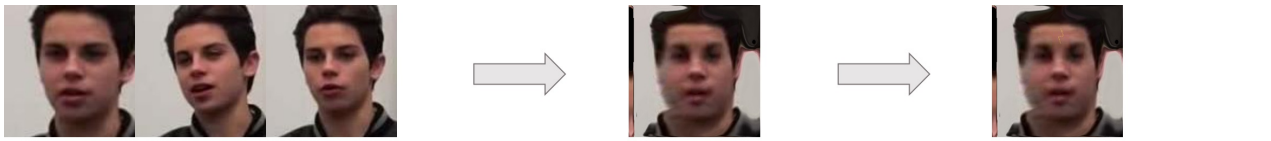
(d)

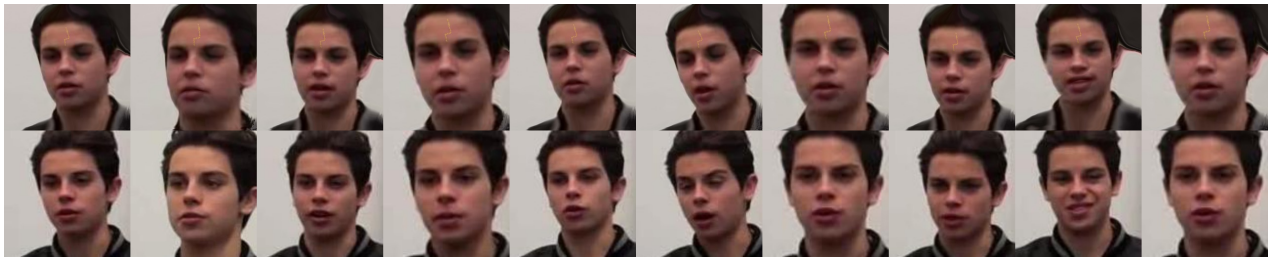
Fig. 13: Additional results of the video editing application using the Harry Potter scar. (a) For the given *source* frames, the *embedded* face is extracted and modified. (b) The modified *embedded* face is controlled using a sequence of *driving* frames (bottom) and the result is shown (top). Best viewed in colour. Zoom in for details.





(b)





(d)

Fig. 14: Additional results of the video editing application using the Harry Potter scar. (a) For the given *source* frames, the *embedded* face is extracted and modified. (b) The modified *embedded* face is controlled using a sequence of *driving* frames (bottom) and the result is shown (top). Best viewed in colour. Zoom in for details.



Generated frames with varying yaw, pitch and roll angle (top row, middle row and bottom row) using pose code vectors to drive the frame generation.

Fig. 15: Controlling the three head pose angles. This example can also be found in the accompanying video.

Chapter 9

SynSin: End-to-end View Synthesis from a Single Image

This work was presented as an oral and published at CVPR 2020.

This work introduces a model to perform single image view synthesis on real images. Given an image of a scene and new viewpoint, the task is to generate a new image of the scene at that viewpoint. This work introduces a differentiable point cloud renderer to render the predicted 3D structure, and a refinement network to hallucinate missing regions. Our model is trained end to end using a photometric loss in a self-supervised manner; the training data consists of pairs of images and their corresponding change in viewpoint.

SynSin: End-to-end View Synthesis from a Single Image

Olivia Wiles[1]* Georgia Gkioxari[2] Richard Szeliski[3] Justin Johnson[2,4]

[1]University of Oxford [2]Facebook AI Research [3]Facebook [4]University of Michigan



Figure 1: **End-to-end view synthesis.** Given a *single* RGB image (red), SynSin generates images of the scene at new viewpoints (blue). SynSin predicts a 3D point cloud, which is projected onto new views using our differentiable renderer; the rendered point cloud is passed to a GAN to synthesise the output image. SynSin is trained end-to-end, without 3D supervision.

Abstract

View synthesis allows for the generation of new views of a scene given one or more images. This is challenging; it requires comprehensively understanding the 3D scene from images. As a result, current methods typically use multiple images, train on ground-truth depth, or are limited to synthetic data. We propose a novel end-to-end model for this task using a single image *at test time; it is trained on real images without any ground-truth 3D information. To this end, we introduce a novel differentiable point cloud renderer that is used to transform a latent 3D point cloud of features into the target view. The projected features are decoded by our refinement network to inpaint missing regions and generate a realistic output image. The 3D component inside of our generative model allows for interpretable manipulation of the latent feature space at test time,* e.g. *we can animate trajectories from a single image. Additionally, we can generate high resolution images and generalise to other input resolutions. We outperform baselines and prior work on the Matterport, Replica, and RealEstate10K datasets.*

1. Introduction

Given an image of a scene, as in Fig. 1 (top-left), what would one see when turning left or walking forward? We can reason that the window and the wall will extend to the left and more chairs will appear to the right. The task of novel *view synthesis* addresses these questions: given a view of a scene, the aim is to generate images of the scene from new viewpoints. This task has wide applications in image editing, animating still photographs or viewing RGB images in 3D. To unlock these applications for any input image, our goal is to perform view synthesis in *complex, real-world scenes* using only *a single input image*.

View synthesis is challenging, as it requires comprehensive scene understanding. Specifically, successful view synthesis requires understanding both the **3D structure** and the **semantics** of the input image. Modelling 3D structure is important for capturing the relative motion of visible objects under a view transform. For example in Fig. 1 (bottom-left), the sink is closer than the shower and thus shifts more as we change viewpoints. Understanding semantics is necessary for synthesising plausible completions of partially visible objects, *e.g.* the chair in Fig. 1 (top-left).

One way to overcome these challenges is to relax the single-image constraint and use *multiple* views to reconstruct 3D scene geometry [12, 15, 51, 76]. This also simplifies semantic modelling, as fewer positions will be occluded from all views. Recent methods [56, 68, 74] can be extremely effective even for complex real-world scenes. However the assumption of multiple views severely limits their applicability, since the vast majority of images are not accompanied by views from other angles.

Project page: `www.robots.ox.ac.uk/~ow/synsin.html`.

*Work done during an internship at Facebook AI Research.

Another approach is to train a convolutional network to estimate depth from images [13, 33], enabling single-image view synthesis in realistic scenes [40]. Unfortunately this approach requires a training dataset of images with ground-truth depth. Worse, depth predictors may not generalise beyond the scene types on which they are trained (*e.g.* a network trained on indoor scenes will not work on outdoor images) so this approach can only perform view synthesis on scene types for which ground-truth depth can be obtained.

To overcome these shortcomings, there has been growing interest in view synthesis methods that do not use any 3D information during training. Instead, an end-to-end generative model with 3D-aware intermediate representations can be trained from image supervision alone. Existing methods have shown promise on synthetic scenes of single objects [31, 54, 55, 60, 67], but have been unable to scale to complex real-world scenes. In particular, several recent methods represent 3D structure using dense voxel grids of latent features [36, 54]. With voxels, the fidelity of 3D information that can be represented is tied to the voxel dimensions, thus limiting the output resolution. On the other hand, point clouds are more flexible, generalise naturally to varying resolutions and are more efficient.

In this paper we introduce SynSin, a model for view **syn**thesis from a **sin**gle image in complex real-world scenes. SynSin is an end-to-end model trained without any ground-truth 3D supervision. It represents 3D scene structure using a high-resolution point cloud of learned features, predicted from the input image using a pair of convolutional networks. To generate new views from the point cloud, we render it from the target view using a high-performance differentiable point cloud renderer. SynSin models scene semantics by building upon recent advances in generative models [3], and training adversarially against learned discriminators. Since all model components are differentiable, SynSin is trained end-to-end using image pairs and their relative camera poses; at test-time it receives only a single image and a target viewpoint.

We evaluate our approach on three complex real-world datasets: Matterport [4], RealEstate10K [74], and Replica [58]. All datasets include large angle changes and translations, increasing the difficulty of the task. We demonstrate that our approach generates high-quality images and outperforms baseline methods that use voxel-based 3D representations. We also show that our trained models can generalise at test-time to high-resolution output images, and even to new datasets with novel scene types.

2. Related work

Research into new view synthesis has a long history in computer vision. These works differ based on whether they use multiple images or a single image at test time and on whether they require annotated 3D or semantic information.

View synthesis from multiple images. If multiple images of a scene can be obtained, inferred 3D geometry can be used to reconstruct the scene and then generate new views. Traditionally, this was done using depth maps [5, 47] or multi-view geometry [11, 12, 15, 30, 51, 76].

In the learning era, DNNs can be used to learn depth. [1, 9, 23, 36, 38, 41] use a DNN to improve view synthesis from a set of noisy, incomplete, or inconsistent depth maps. Given two or more images of a scene within a small baseline, [16, 56, 57, 63, 68, 74] show impressive results at synthesising views within this narrow baseline. [35, 42, 54] learn an implicit voxel representation of one object given many training views and generate new views of that object at test time. [14] use no implicit 3D representation. Unlike these methods, we assume only one image at test time.

View synthesis from a single image using ground-truth depth or semantics. A second vein of work assumes a large dataset of images with corresponding ground-truth 3D and semantic information to train their 3D representation [40, 52, 62]. These methods are reliant on a large scale benchmark and corresponding annotation effort. The depth may be obtained using a depth or lidar camera [17, 28, 53] or SfM [33]; however, this is time-consuming and challenging, especially for outdoor scenes, often necessitating the use of synthetic environments. We aim to make predictions anywhere, *e.g.* the wood scene in Fig. 5, and in realistic settings, *without* 3D information or semantic labels.

View synthesis from a single image. DNNs can be used to learn view synthesis in an end-to-end fashion. One such line of work synthesises new views using purely image to image transformations [7, 31, 43, 59, 60, 75]. Later work performs 3D operations directly on the learned embedding [67] or interprets the latent space as an implicit surface [55]. However, these works consider synthetic datasets with a single object per image and train one model per object class. Most similar to ours is the recent work of [8]. However, they do not consider larger movements that lead to significant holes and dis-occlusions in the target image. They also consider a more constrained setup; they consider synthetic object classes and mostly forward motion in KITTI [17], whereas we use a variety of indoor and outdoor scenes.

Many works explore using a DNN to predict 3D object shapes [18, 20, 24, 26, 64, 69] or the depth of a scene given an image [6, 13, 33, 73]. These works focus on the quality of the 3D predictions as opposed to the view-synthesis task.

Generative models. We build on recent advances in generative models to produce high-quality images with DNNs [3, 19, 27, 39, 44]. In [3, 27], moving between the latent codes of different instances of an object class seemingly interpolates pose, but explicitly modifying pose is hard to control and evaluate. [39] allows for explicit pose control but not from a given image; they also use a voxel representation, which we find to be computationally limiting.

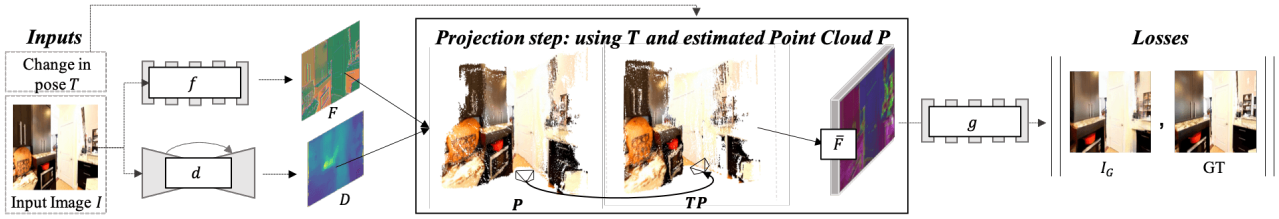


Figure 2: **Our end-to-end system.** The system takes as input an image I of a scene and change in pose $\mathtt{T}$. The *spatial feature predictor* (f) learns a set of features F (visualised by projecting features using PCA to RGB) and the *depth regressor* (d) a depth map D. F are projected into 3D (the diagram shows RGB for clarity) to give a point cloud $\mathcal{P}$ of features. $\mathcal{P}$ is transformed according to $\mathtt{T}$ and rendered. The rendered features $\bar{F}$ are passed through the *refinement network* (g) to generate the final image I_G. I_G should match the target image, which we enforce using a set of discriminators and photometric losses.

3. Method

In this section, we introduce SynSin (Fig. 2) and in particular how we overcome the two main challenges of representing 3D scene structure and scene semantics. To represent the 3D scene structure, we project the image into a latent feature space which is in turn transformed using a differentiable point cloud renderer. This renderer injects a 3D prior into the network, as the predicted 3D structure must obey geometric principles. To satisfy the scene semantics, we frame the entire end-to-end system as a GAN and build on architectural innovations of recent state-of-the-art generative models.

SynSin takes an input image I and relative pose $\mathtt{T}$. The input image is embedded to a feature space F via a *spatial feature predictor* (f), and a depth map D via a *depth regressor* (d). From F and D, a point cloud $\mathcal{P}$ is created which is rendered into the new view (*neural point cloud renderer*). The *refinement network* (g) refines the rendered features to give the final generated image I_G. At training time, we enforce that I_G should match the target image (*discriminator*).

3.1. Spatial feature and depth networks

Two networks, f and d, are responsible for mapping the raw input image into a higher dimensional feature map and a depth map, respectively. The spatial feature network predicts feature maps at the same resolution as the original image. These feature maps should represent scene semantics, *i.e.* a higher-level representation than simply RGB colours. The depth network estimates the 3D structure of the input image at the same resolution. The depth does not have to be (nor would we expect it to be) perfectly accurate; however, it is explicitly learned in order to perform the task. The design for f and d follows standard architectures built for the two tasks respectively:

Spatial feature network f. We build on the BigGAN architecture [3] and use 8 ResNet blocks that maintain image resolution; the final block predicts a 64-dimensional feature for each pixel of the input image.

Depth network d. We use a UNet [48] with 8 downsampling and upsampling layers to give a final prediction of the same spatial resolution as the input. This is followed by a sigmoid layer and a renormalisation step so the predicted depths fall within the per-dataset min and max values. Please refer to the supplement for the precise details.

3.2. Neural point cloud renderer

We combine the spatial features F and predicted depths D to give a 3D point cloud of feature vectors $\mathcal{P}$. Given the input view transform $\mathtt{T}$, we want to view this point cloud at the target viewpoint. This requires rendering the point cloud. Renderers are used extensively in graphics, as reviewed in [29, 49], but they usually focus on forward projection. Our 3D renderer is a component of an end-to-end system, which is jointly optimised, and so needs to allow for gradient propagation; we want to train for depth prediction *without* any 3D supervision but only with a loss on the final rendered image. Additionally, unlike traditional rendering pipelines, we are not rendering RGB colours but *features*.

Limitations of a naïve renderer. A naïve renderer projects 3D points p_i to one pixel or a small region – the *footprint* – in the new view. Points are sorted in depth using a z-buffer. For all points in the new view, the nearest point in depth (using the z-buffer) is chosen to colour that point. A non-differentiable renderer does not provide gradients with respect to the point cloud positions (needed to train our depth predictor) nor the feature vectors (needed to train our spatial feature network). Simply making the operations of a naïve renderer differentiable is problematic for two reasons (illustrated in Fig. 3). **(1) Small neighbourhoods:** each point projects to only one or a few pixels in the rendered view. In this case, there are only a few gradients for each point in the xy-plane of the rendered view; this drawback of *local* gradients is discussed in [25] in the context of bilinear samplers. **(2) The hard z-buffer:** each rendered pixel is only affected by the nearest point in the z-buffer (*e.g.* if a new pixel becomes closer in depth, the output will suddenly change).

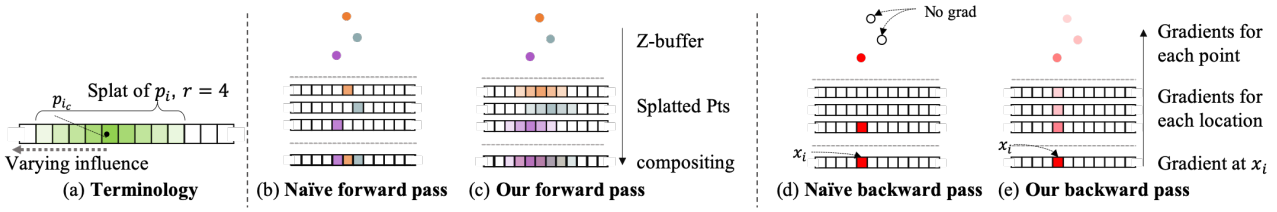


Figure 3: Comparison of our rendering pipeline to a naïve version. Given a set of points ordered in a z-buffer, our renderer projects points to a region of radius r using α-compositing, not just the nearest point. When back-propagating through our renderer, gradients flow not just to the nearest point, but to all points in the z-buffer. (For simplicity we show 1D projections.)

Our solution. We propose a neural point cloud renderer in order to solve the prior two problems by softening the hard decisions, as in Fig. 3. This is inspired by [34], which introduces a differentiable renderer for meshes by similarly softening the hard rasterisation decisions and [77] which renderers point clouds by splatting points to a region and accumulating. First, to solve the issue of small neighbourhoods, we splat 3D points to a disk of varying influence controlled by hyperparameters r and M. Second, to solve the issue of the hard z-buffer, we accumulate the effects of the K nearest points, not just the nearest point, using a hyperparameter γ.

Our renderer first projects $\mathcal{P}$ onto a 2D grid under the given transformation $\mathtt{T}$. A 3D point p_i is projected and splatted to a region with centre p_{i_c} and radius r. The influence of the 3D point p_i on a pixel l_{xy} is proportional to the Euclidean distance $d_2(\cdot,\cdot)$ from the centre of the region:

$$\mathcal{N}(p_i, l_{xy}) = 0 \qquad \text{if} \quad d_2(p_{i_c}, l_{xy}) > r$$
$$\mathcal{N}(p_i, l_{xy}) = 1 - \frac{d_2(p_{i_c}, l_{xy})}{M} \qquad \text{otherwise.}$$

Though $\mathcal{N}$ is not differentiable, we can approximate derivatives using the subderivative. r and M control the spread and fall-off of the influence of a 3D point.

The projected points are then accumulated in a z-buffer; they are sorted according to their distance d_i from the new camera and only the K nearest points kept for each pixel in the new view. The sorted points are accumulated using alpha over-compositing (where γ is a hyperparameter):

$$\rho_{i_{mn}} = \mathcal{N}(p_i, l_{mn}) \tag{1}$$
$$\bar{F}_{mn} = \sum_{i=1}^{K} \rho_{i_{mn}}^{\gamma} F_i \prod_{j=1}^{i-1} (1 - \rho_{j_{mn}}^{\gamma}), \tag{2}$$

where $\bar{F}$ is the projected feature map in the new view and F in the original view. γ controls the blending; if $\gamma = 0$, this is hard z-buffering. This setup is illustrated in Fig. 3.

Implementation. Our renderer must be high-performance, since we process batches of high-resolution point clouds during training. We implement our renderer using a sequence of custom CUDA kernels, building upon work on high-performance triangle rasterisation with CUDA [32]. We use a two-stage approach: in the first stage we break the output image into tiles, and determine the set of points whose footprint intersects each tile. In the second stage, we determine the K nearest points for each pixel in the output image, sorting points in depth using per-pixel priority queues in shared memory to reduce global memory traffic.

Other approaches. This method is related to the point cloud rasterisers of [1, 24, 70]. However, our renderer is a simpler than [70] and we apply it in an end-to-end framework. While [1] also renders point clouds of features, they only back-propagate to the feature vectors, not the 3D positions. [24] stores the predicted points in a voxel grid before performing the projection step; this limits the resolution.

Performance. On a single V100 GPU, rendering a batch of six point clouds with $512^2 = 262{,}144$ points each to a batch of six images of size 256×256 takes 36ms for the forward pass and 5ms for the backward pass. In contrast, converting the same point cloud to a 256^3 voxel grid using the implementation from [24] takes $\approx$ 1000ms for the forward pass and $\approx$ 2000ms for the backward pass.

3.3. Refinement module and discriminator

Even if the features are projected accurately, regions not visible in the input view will be empty in the target view. The refinement module should inpaint [2, 10] these missing regions in a semantically meaningful (*e.g.* missing portions of a couch should be filled in with similar texture) and geometrically accurate (*e.g.* straight lines should continue to be straight) manner. To solve this task, we take inspiration from recent generative models [3, 27, 44].

Deep networks have been previously applied to inpainting [46, 61, 66]. In a typical inpainting setup, we know a-priori which pixels are correct and which need to be synthesised. In our case, the refinement network should perform two tasks. First, it should inpaint regions with no projected features, *e.g.* regions on the image boundary or dis-occluded regions. The refinement module can discover these regions, as the features have values near zero. Second, the refinement module should correct local errors (*e.g.* noisy regions resulting from noisy depth).



Figure 4: Qualitative results on RealEstate for ours and baseline methods. Given the input view and the camera parameters, the methods are tasked to produce the target image. The red squares denote interesting differences between the methods. In the upper row, our model better recreates the true 3D; in the bottom row, our model is better able to preserve detail.

To build the refinement network, we use 8 ResNet [21] blocks, taking inspiration from [3]. Unlike [3], we aim to generate a new image conditioned on an input view not a random vector. Consequently, we find that it is important to maintain the image resolution as much as possible to obtain high quality results. We modify their ResNet block to create a downsampling block. The downsampling block is used to decrease the image resolution by two sizes before upsampling to the original image resolution. To model the ambiguity in the inpainting task, we use batch normalisation injected with noise [3]. We additionally apply spectral normalisation following each convolutional layer [71].

The GAN architecture and objective used is that of [65]. We use 2 multi-layer discriminators at a lower and higher resolution and a feature matching loss on the discriminator.

3.4. Training

Training objective. The network is trained with an L1 loss, content loss and discriminator loss between the generated and target image. The total loss is then $\mathcal{L} = \lambda_{GAN}\mathcal{L}_{GAN} + \lambda_{l1}\mathcal{L}_{l1} + \lambda_c\mathcal{L}_c$.

Training details. The models are trained with the Adam optimiser using a 0.01 learning rate for the discriminator, 0.0001 for the generator and momentum parameters (0, 0.9). $\lambda_{GAN} = 1$, $\lambda_c = 10$, $\lambda_{l1} = 1$. $\gamma = 1$, $r = 4$ pixels, $K = 128$, $W = H = 256$. The models are trained for 50K iterations. We implement our models in PyTorch [45]; they take 1-2 days to train on 3 Tesla V100 GPUs.

4. Experiments

We evaluate our approach on the task of view synthesis using novel real-word scenes. We validate our design choices in Section 4.3 by ablating our approach and comparing against competing end-to-end view synthesis pipelines. We also compare to other systems and find that our model performs better than one based on a trained depth predictor, which fails to generalise well to the new domain. We additionally evaluate SynSin's generalisation performance to novel domains (Section 4.3) as well as higher image resolutions (Section 4.4). Finally, we use SynSin to synthesise trajectories from an initial image in Section 4.6, demonstrating that it can be used for a walk-through application. Additional results are given in the supplement.

4.1. Experimental setup

Datasets. We focus on using realistic data of indoor and outdoor environments as opposed to synthetic objects.

The first framework we use is Habitat [50], which allows for testing in a variety of scanned indoor scenes. The Habitat framework can efficiently generate image and viewpoint pairs for an input scene. We use two sources of indoor scenes: Matterport3D [4], consisting of reconstructions of homes, and Replica [58], which consists of higher fidelity scans of indoor scenes. The Matterport3D dataset is divided at the scene level into train/val/test which contain 61/11/18 scenes. The Replica dataset is only used at evaluation time to test generalisability. Pairs of images are generated by randomly selecting a viewpoint in a scene and then randomly modifying the viewing angle in a range of $\pm 20^\circ$ in each Euclidean direction and the position within ± 0.32m.

The second dataset we use is RealEstate10K [74], which consists of videos of walkthroughs of properties and the corresponding camera parameters (intrinsic and extrinsic) obtained using SfM. The dataset contains both indoor and outdoor scenes. It is pre split into a disjoint set of train and test scenes; we subdivide train into a training and validation set to give approximately 57K/14K/7K scenes in train/val/test. The scenes in the test set are unseen. We sample viewpoints by selecting a reference video frame and then selecting a second video frame a maximum of 30 frames apart. In order to sample more challenging frames, we choose pairs

	Matterport [4]									RealEstate10K [74]			Replica [58]		
	PSNR ↑			SSIM ↑			Perc Sim ↓			PSNR ↑	SSIM ↑	Perc Sim ↓	PSNR ↑	SSIM ↑	Perc Sim ↓
	Both	InVis	Vis	Both	InVis	Vis	Both	InVis	Vis						
1. SynSin (small ft)	**21.14**	**20.19**	**21.84**	**0.71**	0.70	0.69	**1.68**	0.45	**0.98**	$21.10_{3.48}$	$0.73_{0.14}$	$1.34_{0.55}$	**22.36**	0.80	1.64
2. SynSin (hard z)	21.08	20.23	21.70	0.70	0.70	0.67	1.82	0.44	1.11	$21.40_{4.06}$	$0.70_{0.15}$	$1.45_{0.61}$	20.70	0.76	1.95
3. SynSin (rgb)	20.64	19.87	21.21	0.67	0.69	0.65	2.06	0.49	1.27	$20.92_{3.81}$	$0.68_{0.14}$	$1.67_{0.51}$	20.44	0.75	2.03
4. SynSin	20.91	19.80	21.62	**0.71**	**0.71**	**0.70**	**1.68**	**0.43**	**0.99**	$\mathbf{22.31}_{4.97}$	$\mathbf{0.74}_{0.16}$	$\mathbf{1.18}_{0.64}$	21.94	**0.81**	**1.55**
5. SynSin (w/ GT)	22.65	19.64	26.19	0.78	0.71	0.82	1.37	0.50	0.64	–	–	–	23.72	0.86	1.22
6. SynSin (sup. by GT)	21.59	20.32	22.46	0.72	0.71	0.71	1.60	0.43	0.92	–	–	–	22.54	0.80	1.55
7. Im2Im	15.87	16.20	15.97	0.53	0.60	0.48	2.99	0.58	2.05	$17.05_{4.78}$	$0.56_{0.18}$	$2.19_{1.22}$	17.42	0.66	2.29
8. Vox w/ UNet	18.52	17.85	19.05	0.57	0.57	0.57	2.98	0.77	1.96	$17.31_{2.63}$	$0.53_{0.15}$	$2.30_{0.40}$	18.69	0.71	2.68
9. Vox w/ ours	20.62	19.64	21.22	0.70	0.69	0.68	1.97	0.47	1.19	$21.88_{4.39}$	$0.71_{0.15}$	$1.30_{0.55}$	19.77	0.75	2.24

Table 1: Results on Matterport3D [4], RealEstate10K [74], and Replica [58]. ↑ denotes higher is better, ↓ lower is better. XX_{YY} denotes std dev. YY. The ablations demonstrate the utility of each aspect of our model. We outperform all baselines for both datasets and are nearly as good as a model supervised with depth (SynSin (sup. by GT)). We also perform best when considering regions visible (Vis) and not visible (InVis) in the input view.

with a change in angle of $> 5^\circ$ and a change in position of greater than 0.15 if possible (see [74] for a discussion on metric scale). To report results, we randomly generate a set of 2000 pairs of images from the test set.

Metrics. Determining the similarity of images in a manner correlated with human judgement is challenging [72]. We report multiple metrics to obtain a more robust estimate of the relative quality of images. We report the PSNR, SSIM, and perceptual similarity of the images generated by the different models. Perceptual similarity has been recently demonstrated to be an effective method for comparing the similarity of images [72]. Finally, we validate that these metrics do indeed correlate with human judgement by performing a user study on Amazon Mechanical Turk (AMT).

4.2. Baselines

We first abate the need for a soft differentiable renderer by comparing to variants with a small footprint, hard z-buffering, and that directly project RGB values. These models use the same setup, training schedule, and sequence of input images/viewpoints as SynSin.

SynSin (small ft): We set $K = 128$ and $r = 0.5$ in our model to investigate the utility of a large footprint.

SynSin (hard z): We set $K = 1$ and $r = 4$ in our model to investigate the utility of the soft z-buffer.

SynSin (rgb): We project RGB values not features.

SynSin does not assume ground-truth depth at test time; the depth predictor is trained end-to-end for the given task. We investigate the impact of ground-truth (GT) depth by reporting two variants of our model. These models act as upper bounds and can only be trained on Matterport3D (not RealEstate10K), as they use true depth information.

SynSin (w/ GT): The true depth is used as D.

SynSin (sup. by GT): D is supervised by the true depth. (In all other cases SynSin's D is learned with *no* supervision).

We evaluate our 3D representation by comparing to a method that uses no 3D and one that uses voxels. As no methods exist for the challenging datasets we consider, we re-implement the baselines for a fair comparison. The baselines use the same setup, training schedule, and sequence of input images/viewpoints as SynSin.

Im2Im: This baseline evaluates an image-to-image method; we re-implement [75]. [75] only considered a set of discretised rotations about the azimuth and a smaller set of rotations in elevation. However, the changes in viewpoint in our datasets arise from rotating continuously in any direction and translating in 3D. We modify their method to allow for these more complex transformations.

Vox: This baseline swaps our implicit 3D representation for a voxel based representation. The model is based on that of [54]. However, [54] trains one model per object, so their model effectively learns to interpolate between the $>$100 training views unlike our model, which extrapolates to new real-world test scenes given a *single* input view. We consider two variants: **Vox w/ UNet** uses the UNet encoder/decoder of [54] whereas **Vox w/ ours** uses a similar ResNet encoder/decoder setup to SynSin. This comparison evaluates our 3D approach as opposed to a voxel based one as well as whether our encoder/decoder setup is preferable.

Finally, we compare SynSin to existing pipelines that perform view synthesis. These systems make different assumptions and follow different approaches. This comparison validates our use of a learned end-to-end system.

StereoMag [74]: This system takes two images as input at test time. Assuming two input views simplifies the problem of 3D understanding compared to our work, which estimates 3D from a single view.

3D View: This system trains a single-image depth predictor on images with ground-truth depth (*e.g.* MegaDepth [33]). Predicted depths are used to convert the input image to a textured 3D mesh, which is extended in space near occlusion boundaries using isotropic colour diffusion [22]. Finally the mesh is rendered from the target view. The approach is similar to 3D Photos [37].

System comparison on RealEstate10K [74]			
	PSNR ↑	SSIM ↑	Perc Sim ↓
SynSin	$22.31_{4.97}$	$0.74_{0.16}$	$1.18_{0.64}$
3DView	$21.88_{8.43}$	$0.66_{0.22}$	$1.52_{1.03}$
StereoMag [74]	$25.34_{9.48}$	$0.82_{0.13}$	$1.19_{0.77}$

Table 2: SynSin performs better than a system trained with GT depth (3DView) and approaches the performance of [74], which uses 2 input views at test time.

	Generalisation to higher res.		
	PSNR ↑	SSIM ↑	Perc Sim ↓
SynSin	$\mathbf{22.06}_{6.30}$	$\mathbf{0.72}_{0.18}$	$\mathbf{1.00}_{0.65}$
Vox w/ ours	$18.82_{2.52}$	$0.61_{0.14}$	$2.47_{0.36}$

Table 3: Results when applying models trained on 256×256 images to 512×512 images.

AMT User Study			
	Ours	Vox w/ ours	Neither
E-O	**68.7**	31.3	–
E-O-N	**55.6**	27.3	17.2

Table 4: % of videos chosen as most realistic. In *E-O*, users choose the better method; in *E-O-N*, users can say neither is better.



Figure 5: System comparisons on RealEstate10K, illustrating failure cases. Note StereoMag [74] uses two input images (second is shown as an inset). Unlike [74] we inpaint missing regions (bottom row); [74] fails to model the left region and cannot inpaint the missing region. 3DView uses a model pretrained for depth, causing their system to produce inaccurate results in some cases (*e.g.* the bed in the top row).

4.3. Comparisons with other methods

Results on Matterport3D and RealEstate10K. We train our models, ablations, and baselines on these datasets.

To better analyse the results, we compare models on how well they understand the 3D scene structure and the scene semantics (discussed in Section 1). To achieve this, we report metrics on the final prediction (Both) but also on the regions of the target image that are visible (Vis) and not visible (InVis) in the input image. (Vis) evaluates the quality of the learned 3D scene structure, as it can be largely solved by accurate depth prediction. (InVis) evaluates the quality of a model's understanding of scene semantics; it requires a holistic understanding of semantic and geometric properties to reasonably in-paint missing regions. In order to determine the (Vis) and (InVis) regions, we use the GT depth in the input view to obtain a binary mask of which pixels are visible in the target image. This is only possible on Matterport3D (RealEstate10K does not have GT depth).

Table 1 and Fig. 4 report results on Matterport3D and RealEstate10K. On both datasets, we perform better than the baselines on all metrics and under all conditions, demonstrating the utility of both our 3D representation and our inpainting module. These results demonstrate that the differentiable renderer is important for training the depth model (rows 1-4). Our encoder decoder setup is shown to be important, as it improves the baseline's performance significantly (rows 8-9). Qualitatively, our model preserves fine detail and predicts 3D structure better than the baselines.

System comparison on RealEstate10K. We compare our system to the 3DView and StereoMag [74] in Table 2 and Fig. 5. Our model performs better than 3DView despite their method having been trained with hundreds of thousands of depth images. We hypothesise that this gap in performance is due to the 3DView's depth prediction not generalising well; their dataset consists of images of mostly close ups of objects whereas ours consists of scenes taken inside or outdoors. This baseline demonstrates that using an explicit 3D representation is problematic when the test domain differs from the training domain, as the depth predictor cannot generalise. Finally, our method of inpainting is better than that of 3DView, which produces a blurry result. [74] does not inpaint unseen regions in the generated image.

Comparison with upper bounds. We compare our model to SynSin (w/ GT) and SynSin (sup. with GT) in Table 1. These models either use GT depth or are supervised by GT depth; they are upper bounds of performance. While there is a performance gap between SynSin and SynSin (w/ GT) under the (Vis) condition, this gap shrinks for the (InVis) condition. Interestingly, SynSin trained with *no* depth supervision performs nearly as well as SynSin (sup. with GT) under both the (Vis) and (InVis) conditions; our model also generalises better to the Replica dataset. This experiment demonstrates that having true depth during training does not necessarily give a large boost in a downstream task and could hurt generalisation performance. It validates our decision to use an end-to-end system (as opposed to using depth estimated from a self-supervised method).

Generalisation to Replica. Given the models trained on Matterport3D, we evaluate generalisation performance (with no further fine-tuning) on Replica in Table 1. Replica contains additional types of rooms (*e.g.* office and hotel rooms) and is higher quality than Matterport (it has fewer geometric and lighting artefacts and more complex textures). SynSin generalises better to this unseen dataset; qualitatively, SynSin seems to introduce fewer artefacts (Fig. 6).

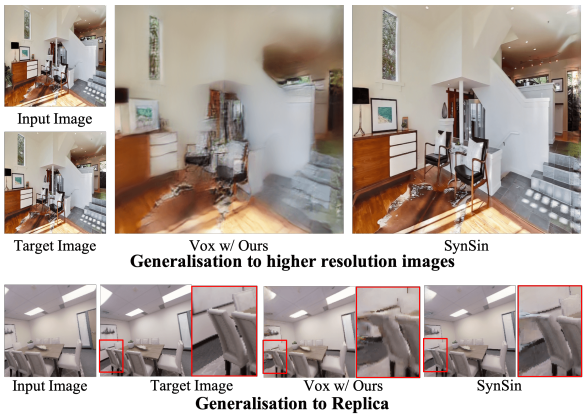


Figure 6: Comparison of SynSin against the baseline, Vox w/ ours, at generalising to higher res 512×512 images and Replica [58]. Ours generalises better with fewer artefacts.

4.4. Generalisation to higher resolution images

We also evaluate generalisation to higher image resolutions in Table 3 and Fig. 6. SynSin can be applied to higher resolution images without any further training. The ability to generalise to higher resolutions is due to the flexible 3D representation in our approach: the networks are fully convolutional and the 3D point cloud can be sampled at any resolution to maintain the resolution of the features. As a result, it is straightforward at test time to apply a network trained on a smaller image size (*e.g.* 256×256) to one of a different size (*e.g.* 512×512). Unlike our approach, the voxel baseline suffers a dramatic performance drop when applied to a higher resolution image. This drop in performance is presumably a result of the heavy downsampling and imprecision resulting from representing the world as a coarse voxel grid.

4.5. Depth predictions

We evaluate the quality of the learned 3D representation qualitatively in Fig. 7 for SynSin trained on RealEstate10K. We note that the accuracy of the depth prediction only matters in so far as it improves results on the view synthesis task. However, we hypothesise that the quality of the generated images and predicted depth maps are correlated, so looking at the quality of the depth maps should give some insight into the quality of the learned models. The depth map predicted by our method is higher resolution and more realistic than the depth map predicted by the baseline methods. Additionally, our differentiable point cloud renderer appears to improve the depth quality over using a hard z-buffer or a smaller footprint. However, we note that small objects and finer details are not accurately recreated. This is probably because these structures have a limited impact on the generated images.

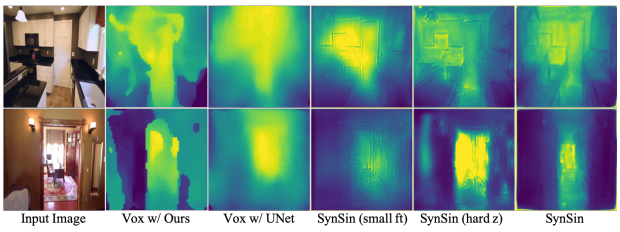


Figure 7: Recovered depth predictions for both our method and the baselines. The baselines predict a less accurate and coarser depth. Using a smaller radius or hard z-buffer produces qualitatively similar or worse depth maps.

4.6. User study: Animating still images

Finally, we task SynSin to synthesise images along a trajectory. Given an initial input frame from a video in RealEstate10K, SynSin generates images at the camera position of the 30 subsequent frames. While changes are hard to see in a figure (*e.g.* Fig. 1), the supplementary videos clearly show smooth motion and 3D effects. These demonstrate that SynSin can generate reasonable videos despite being trained purely on images. To evaluate the quality of the generated videos, we perform an AMT user study.

We randomly choose 100 trajectories and generate videos using SynSin and the Vox w/ ours baseline. Five users are asked to rate which method's video is most realistic. For each video, we take the majority vote to determine the best video. We report the percentage of times the users choose a given method in Table 4.

Either-or setup (E-O): Users rate whether the baseline or our generated video is more realistic.

Either-or-neither setup (E-O-N): Users rate whether the baseline or our generated video is more realistic or whether they are equally realistic/unrealistic (*neither*). When taking the majority vote, if their is no majority, *neither* video is said to be more / less realistic

In both cases, users prefer our method, presumably because our videos have smoother motion and fewer artefacts.

5. Conclusion

We introduced SynSin, an end-to-end model for performing single image view synthesis. At the heart of our system are two key components: *first* a differentiable neural point cloud renderer, and *second* a generative refinement module. We verified that our approach can be learned end-to-end on multiple realistic datasets, generalises to unseen scenes, can be applied directly to higher image resolutions, and can be used to generate reasonable videos along a given trajectory. While we have introduced SynSin in the context of view synthesis, we note that using a neural point cloud renderer within a generative model has applications in other tasks.

Acknowledgements The authors thank Johannes Kopf for sharing code, Manolis Savva and Erik Wijmans for help with the Habitat dataset, and Sebastien Ehrhardt, Oliver Groth, and Weidi Xie for feedback on paper drafts.

References

[1] Kara-Ali Aliev, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

[2] Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proc. ACM SIGGRAPH*, 2000.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proc. ICLR*, 2019.

[4] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D data in indoor environments. *International Conference on 3D Vision (3DV)*, 2017. Matterport3D dataset available at `https://niessner.github.io/Matterport/`.

[5] Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 2013.

[6] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016.

[7] Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *NeurIPS*, 2016.

[8] Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. *Proc. ICCV*, 2019.

[9] Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. *Proc. ICCV*, 2019.

[10] A. Criminisi, P. Pérez, and T. Kentaro. Region filling and object removal by exemplar-based image inpainting. *IEEE Transactions on Image Processing*, 2004.

[11] Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. In *Rendering Techniques*. 1998.

[12] P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image- based approach. In *Proc. ACM SIGGRAPH*, pages 11–20, 1996.

[13] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014.

[14] SM Ali Eslami, Danilo Jimenez Rezende, Frederic Besse, Fabio Viola, Ari S Morcos, Marta Garnelo, Avraham Ruderman, Andrei A Rusu, Ivo Danihelka, and Karol Gregor. Neural scene representation and rendering. *Science*, 360(6394), 2018.

[15] A. W. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. *IJCV*, 63(2):141–151, 2005.

[16] John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proc. CVPR*, 2019.

[17] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

[18] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. *Proc. ICCV*, 2019.

[19] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, pages 2672–2680, 2014.

[20] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *Proc. CVPR*, 2018.

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. CVPR*, 2016.

[22] Peter Hedman and Johannes Kopf. Instant 3d photography. *ACM Transactions on Graphics (TOG)*, 2018.

[23] Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 2018.

[24] Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *NeurIPS*, 2018.

[25] Wei Jiang, Weiwei Sun, Andrea Tagliasacchi, Eduard Trulls, and Kwang Moo Yi. Linearized multi-sampling for differentiable image transformation. *Proc. ICCV*, 2019.

[26] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *Proc. ECCV*, 2018.

[27] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proc. CVPR*, 2019.

[28] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (TOG)*, 2017.

[29] Leif Kobbelt and Mario Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics*, 2004.

[30] Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)*, 2013.

[31] Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *NeurIPS*, 2015.

[32] Samuli Laine and Tero Karras. High-performance software rasterization on gpus. In *Proc. ACM SIGGRAPH Symposium on High Performance Graphics.*, 2011.

[33] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proc. CVPR*, 2018.

[34] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised

single-view mesh reconstruction. *Proc. ICCV*, 2019.

[35] Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 2019.

[36] Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. Lookingood: enhancing performance capture with real-time neural re-rendering. *ACM Transactions on Graphics (TOG)*, 2018.

[37] Kevin Matzen, Matthew Yu, Jonathan Lehman, Peizhao Zhang, Jan-Michael Frahm, Peter Vajda, Johannes Kopf, and Matt Uyttendaele. Powered by AI: Turning any 2D photo into 3D using convolutional neural nets. https://ai.facebook.com/blog/-powered-by-ai-turning-any-2d-photo-into-3d-using-convolutional-neural-nets/, Feb 2020.

[38] Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proc. CVPR*, 2019.

[39] Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3d representations from natural images. In *Proc. ICCV*, 2019.

[40] Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 2019.

[41] David Novotny, Ben Graham, and Jeremy Reizenstein. PerspectiveNet: A scene-consistent image generator for new view synthesis in real indoor environments. In *NeurIPS*, 2019.

[42] Kyle Olszewski, Sergey Tulyakov, Oliver Woodford, Hao Li, and Linjie Luo. Transformable bottleneck networks. In *Proc. ICCV*, 2019.

[43] Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *Proc. CVPR*, 2017.

[44] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In *Proc. CVPR*, 2019.

[45] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, 2019.

[46] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proc. CVPR*, pages 2536–2544, 2016.

[47] Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 2017.

[48] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Springer, 2015.

[49] Miguel Sainz and Renato Pajarola. Point-based rendering techniques. *Computers & Graphics*, 2004.

[50] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied AI research. In *Proc. ICCV*, 2019.

[51] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. CVPR*, 2006.

[52] Daeyun Shin, Zhile Ren, Erik Sudderth, and Charless Fowlkes. 3D scene reconstruction with multi-layer depth and epipolar transformers. In *Proc. ICCV*, 2019.

[53] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *Proc. ECCV*, 2012.

[54] Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. DeepVoxels: Learning persistent 3D feature embeddings. In *Proc. CVPR*, 2019.

[55] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *NeurIPS*, 2019.

[56] Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proc. CVPR*, 2019.

[57] Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4D rgbd light field from a single image. In *Proc. ICCV*, 2017.

[58] Julian Straub, Thomas Whelan, Lingni Ma, Yufan Chen, Erik Wijmans, Simon Green, Jakob J. Engel, Raul Mur-Artal, Carl Ren, Shobhit Verma, Anton Clarkson, Mingfei Yan, Brian Budge, Yajie Yan, Xiaqing Pan, June Yon, Yuyang Zou, Kimberly Leon, Nigel Carter, Jesus Briales, Tyler Gillingham, Elias Mueggler, Luis Pesqueira, Manolis Savva, Dhruv Batra, Hauke M. Strasdat, Renzo De Nardi, Michael Goesele, Steven Lovegrove, and Richard Newcombe. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

[59] Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *Proc. ECCV*, 2018.

[60] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3D models from single images with a convolutional network. In *Proc. ECCV*, 2016.

[61] Piotr Teterwak, Aaron Sarna, Dilip Krishnan, Aaron Maschinot, David Belanger, Ce Liu, and William T Freeman. Boundless: Generative adversarial networks for image extension. In *Proc. ICCV*, 2019.

[62] Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2D image of a 3D scene. In *Proc. CVPR*, 2018.

[63] Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *Proc. ECCV*, 2018.

[64] Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and

Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proc. CVPR*, 2017.

[65] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional GANs. In *Proc. CVPR*, 2018.

[66] Yi Wang, Xin Tao, Xiaoyong Shen, and Jiaya Jia. Wide-context semantic image extrapolation. In *Proc. CVPR*, 2019.

[67] Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Interpretable transformations with encoder-decoder networks. In *Proc. ICCV*, 2017.

[68] Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)*, 2019.

[69] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NeurIPS*, 2016.

[70] Wang Yifan, Felice Serena, Shihao Wu, Cengiz Öztireli, and Olga Sorkine-Hornung. Differentiable surface splatting for point-based geometry processing. *ACM Transactions on Graphics (TOG)*, 2019.

[71] Han Zhang, Ian Goodfellow, Dimitris Metaxas, and Augustus Odena. Self-attention generative adversarial networks. 2019.

[72] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proc. CVPR*, 2018.

[73] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proc. CVPR*, 2017.

[74] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 2018.

[75] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Proc. ECCV*, 2016.

[76] C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM transactions on graphics (TOG)*, 2004.

[77] Matthias Zwicker, Hanspeter Pfister, Jeroen Van Baar, and Markus Gross. Surface splatting. In *Proc. ACM SIGGRAPH*, 2001.

SynSin: Appendix

We give additional results in Section A, additional architectural details in Section B, additional information about baselines in Section C, and finally information about datasets in Section D. Finally, we discuss some choices that did and did not work in Section E.

A. Additional experimental results

Results on KITTI [17]. We evaluated our model on the KITTI dataset in order to compare with [8]. We trained our model on the KITTI dataset and compared with their pretrained model on a held-out set in Table 5. We achieve similar or better results on all metrics. Additionally, because [8] resamples the input image, it cannot generate pixels unseen in the original view. As shown in Fig. 8 (rows 1,2,6), this causes severe artefacts for backward motion.

We additionally show some failure cases for both methods when the viewpoint change is much larger than the average viewpoint change seen at test time in the bottom two rows.

Additional qualitative results. We give additional qualitative results on RealEstate10K (Fig. 9-10), Replica (Fig. 11), and Matterport3D (Fig. 12). The supplementary video shows sample videos of a model generating images along a given trajectory. We compare SynSin to the baseline (Vox w/ ours); SynSin has smoother motion with fewer artefacts. We also visualise additional depth prediction results in Fig. 13-14.

Comparison on KITTI [17]			
	PSNR ↑	SSIM ↑	Perc Sim ↓
SynSin	16.70	0.52	**2.07**
ContView [8]	**16.90**	**0.54**	2.21

Table 5: Comparison on KITTI to [8]. ↑ denotes higher is better, ↓ lower is better.



Figure 8: Qualitative results on KITTI [74] comparing SynSin to [8]. The bottom two rows demonstrate failure cases due to large viewpoint change. Zoom in for details.



Figure 9: Additional results on RealEstate10K [74]. Zoom in for details.

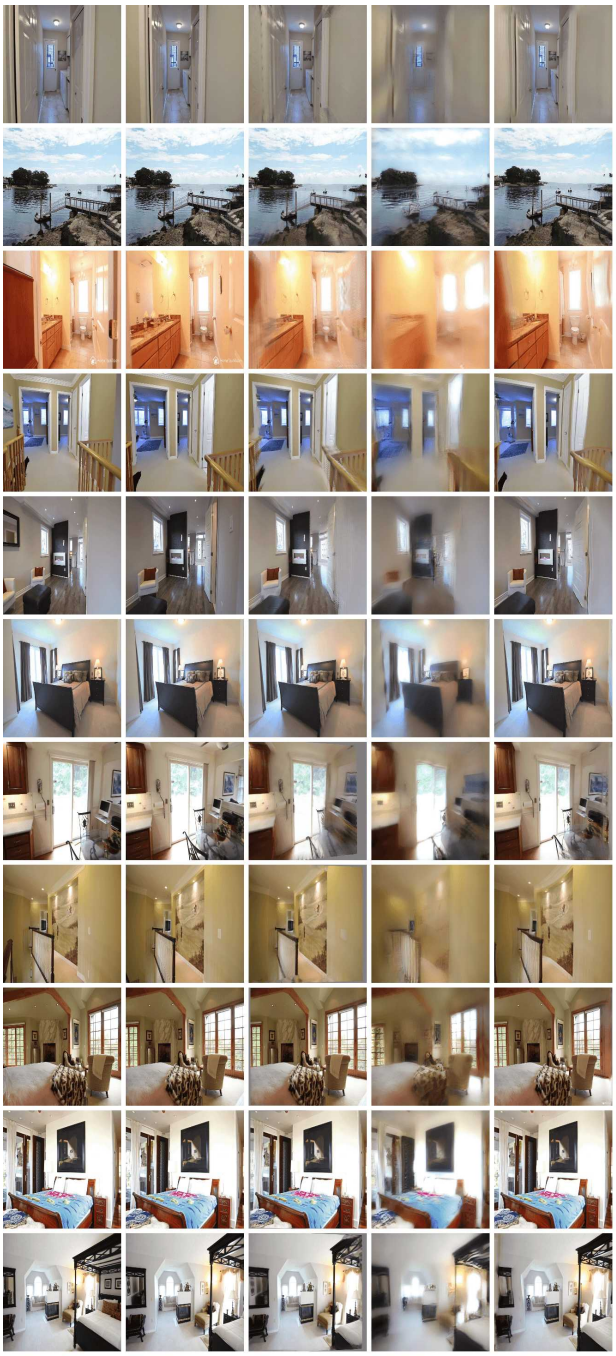


Figure 10: Additional results on RealEstate10K [74]. Zoom in for details.

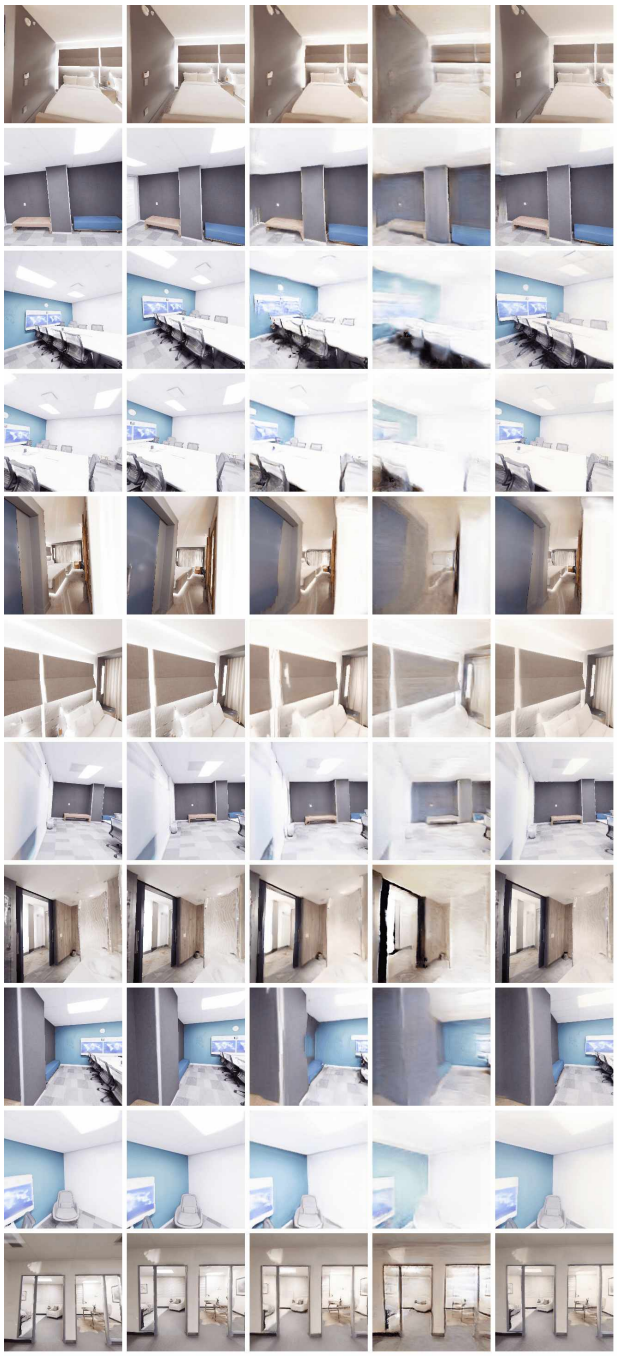


Figure 11: Additional results on Replica [58]. Zoom in for details.

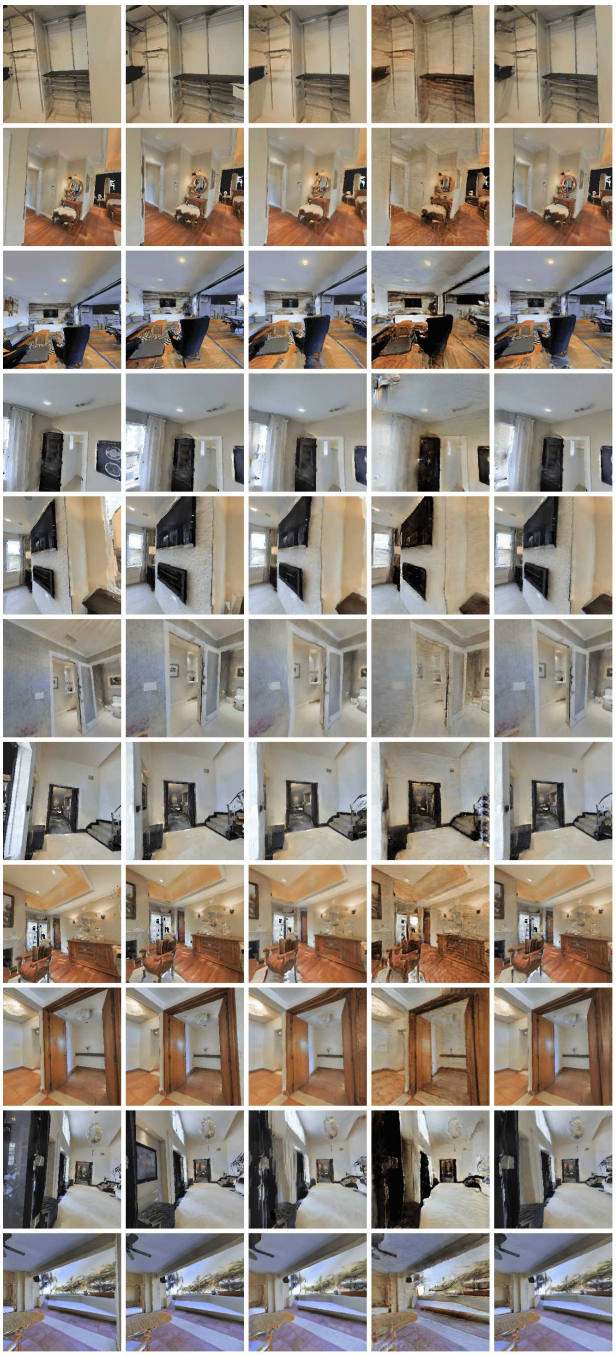


Figure 12: Additional results on Matterport3D [4]. Zoom in for details.

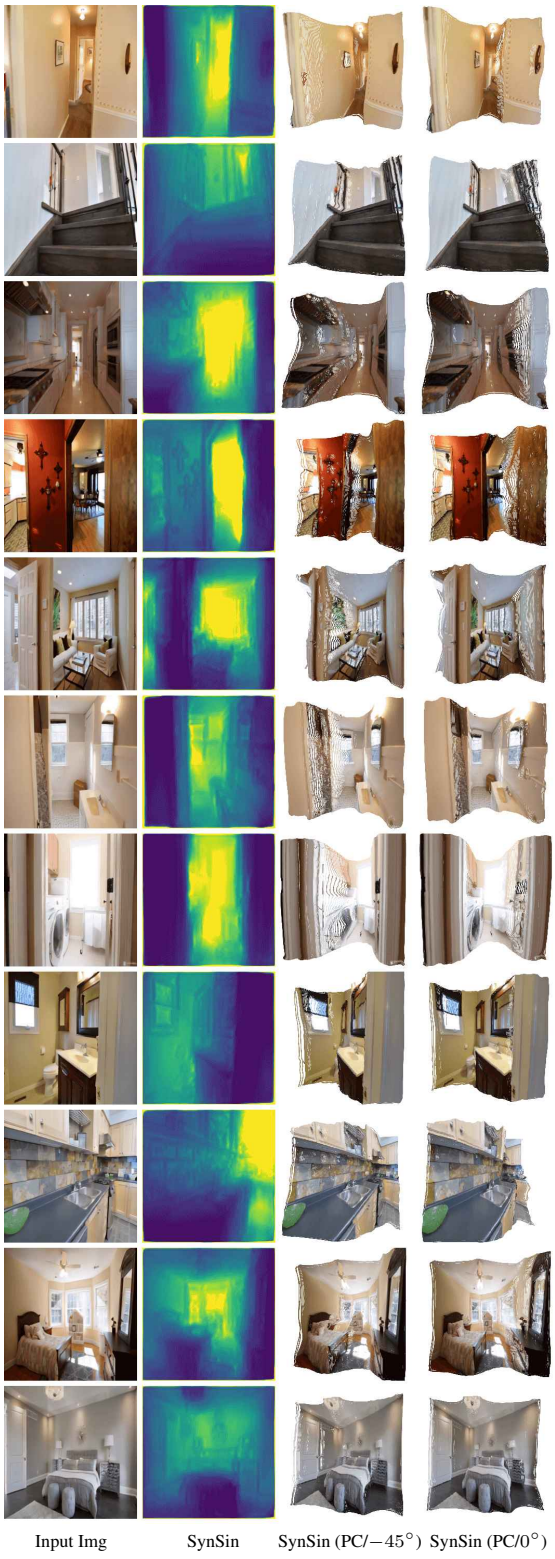


Figure 13: Additional depth predictions on RealEstate10K [74]. We also visualise the point cloud (PC) and the rotated point cloud at $-45°$. (Note that the point cloud in the model is actually a point cloud of features, not RGB values.)

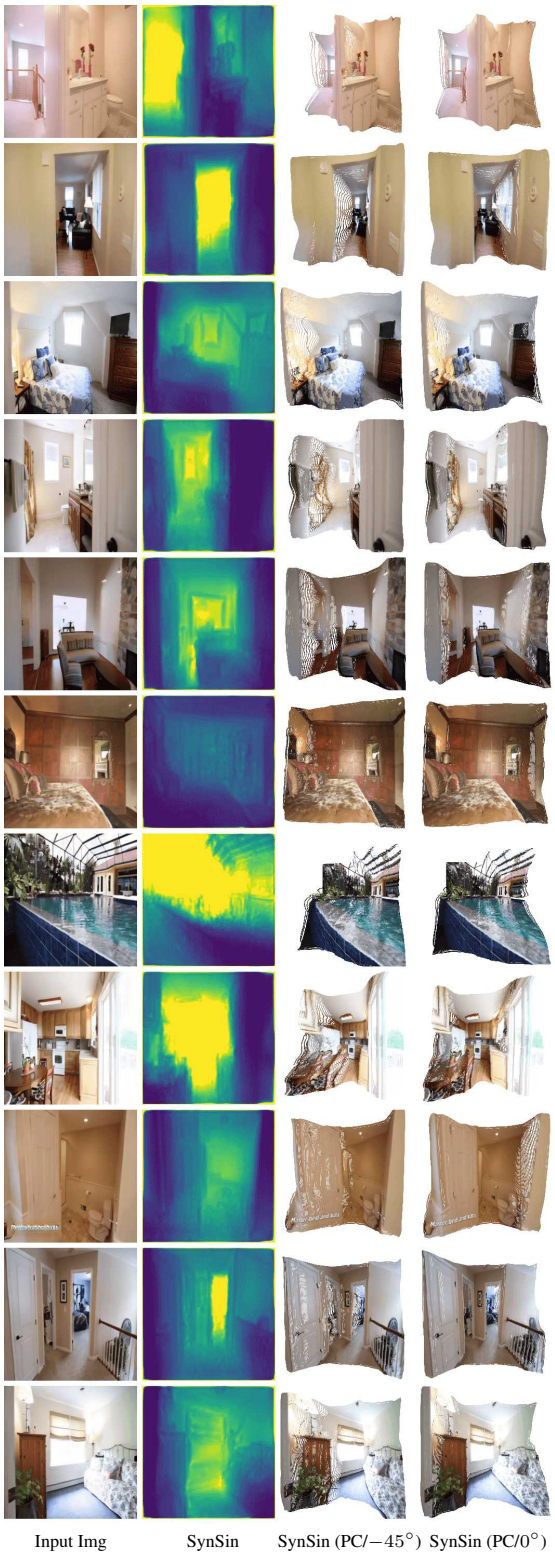


Figure 14: Additional depth predictions on RealEstate10K [74]. We also visualise the point cloud (PC) and the rotated point cloud at $0°$. (Note that the point cloud in the model is actually a point cloud of features, not RGB values.)

B. Additional architectural details

Here we give more information about the precise architectural details used to build the components of our model.

ResNet blocks. Our spatial feature network and refinement networks are composed of ResNet blocks. The ResNet blocks used are the same as those used in [3] (Appendix B, Fig 15 (b)), reproduced in Fig. 15. However, we consider three different setups. The block may be used to increase the resolution of the features using an upsample layer (as used in the original paper by [3]) (Fig. 15(a)). The block may be used to decrease the resolution of the features using an average pooling layer as opposed to the upsample layer (Fig. 15(b)). The block may be used to maintain the resolution of the features using an identity layer as opposed to the upsample layer (Fig. 15(c)).

Spatial feature network. ResNet blocks are stacked together to form the embedding network. In particular, we use the setup in Fig. 16(a).

Refinement network. ResNet blocks are stacked together to form the decoder network. In particular, we use the setup in Fig. 16(b).

Depth regressor. The depth regressor network uses a UNet architecture, as illustrated in Fig. 17.

Additional details on the perceptual loss. We follow the perceptual loss used in [44].

C. Additional details on baselines

In this section, we give further information about the baselines used.

Im to im. We follow the architecture of [75]. However, [75] only considers discrete rotations about the azimuth and a small set of changes in elevation, so [75] takes four values as input, the $\cos$ and $\sin$ values of the azimuth and elevation. However, our datasets include rotation in all three directions, as well as translational motion. As a result, we modify their angle encoder to take 12 values (as opposed to four), and pass the change in viewpoint, $\mathtt{T}$ to the angle encoder. The network is visualised in Fig. 18.

Vox w/ unet. This baseline is based on [54], which represents 3D shape in a neural network using a voxel representation. Note that they train one model per instance, so their model only generalises to that one object. Their overall setup is as follows. An image is passed through an encoder (*e.g.* our spatial feature network) to obtain a set of features. The features are projected into a voxel grid, which is transformed and projected into the new view. The features are accumulated using an occlusion network, which acts as a pseudo depth predictor and predicts the occupancy of the voxels. The predicted occupancy is used to re-weight and combine features. This is then passed to the decoder (*e.g.* our refinement network) which predicts the scene at the new view. Finally, the generated image is compared to the true image using discriminators and photometric losses.

To reimplement this approach, we follow their architectural choices and use a UNet style architecture for all network components (the spatial feature network, refinement network, and occlusion network). However, we use the discriminators and photometric losses used to train SynSin to ensure that both methods are fair in terms of the discriminator. The details for the encoder/decoder setup are given in Fig. 19. The occupancy network is a 3D UNet, which takes as input the rotated voxels and then predicts occupancy for each voxel location; these are then normalised using a softmax layer over the depth dimension. The details are given in Fig. 20. We use their setup but train the network to generate new images of a scene given a *single* image of a scene.

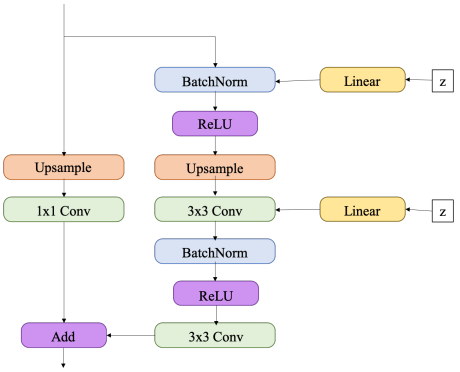
Vox w/ ours. Instead of using the UNet style spatial feature and refinement network in vox w/ ours, we use a sequence of ResNet blocks, as described in Fig. 21. The set of ResNet blocks in the spatial feature network downsamples the image to the appropriate size. The refinement network similarly upsamples the projected features to the appropriate image size. We also use a larger capacity in this setup to ensure that our 3D representation is preferable. The network was trained with a lower learning rate (lr=0.0004) as opposed to (lr=0.001) as in our model, as we found that the model struggled to learn with the higher learning rate.

Other setups. We experimented with other ResNet block sequences and multiple learning rates when creating this baseline. Instead of downsampling the features within the encoder (*e.g.* the spatial feature network), we can use the same spatial feature network as SynSin (to obtain features of size $C \times 256 \times 256$ and then downsample to obtain features of size $C \times 64 \times 64$. Similarly, instead of upsampling the features within the decoder (*e.g.* the refinement network), we can upsample the transformed features to obtain ones of size $C \times 256 \times 256$ and pass these upsampled features to the refinement network and so use the same refinement network we use in SynSin. We found that the results were similar to those of the model used in the paper on RealEstate10K but worse on Matterport.

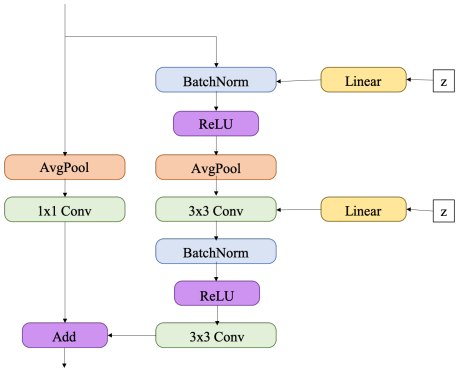
We additionally found that the results were highly dependent on the learning rate for this model.

3DView. This baseline is based on a depth predictor (*e.g.* [33]), so 3DView predicts depth up to a scale ambiguity.

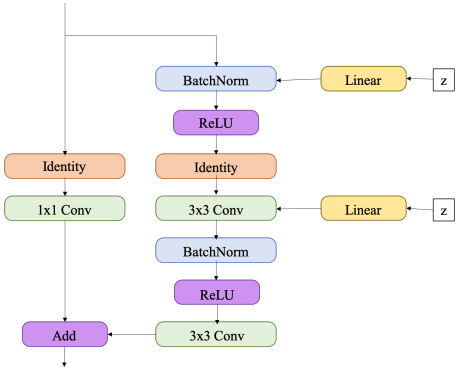
As the depth is only predicted up to a scale, we generate images for multiple possible scales for each test image and then report results for the best image.



(a) ResNet block.

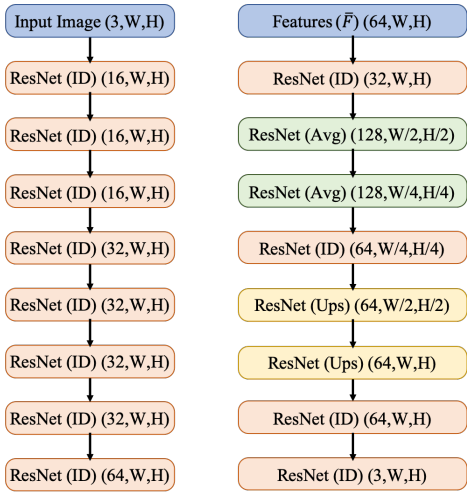


(b) ResNet block with an average pool block.



(c) ResNet block with an identity block.

Figure 15: An overview of ResNet blocks. In (a), we show the basic ResNet block, (b) when we replace the upsample block by an average pool block, and (c) when we replace the upsample block by an identity block.



(a) Spatial feature network.

(b) Refinement network.

Figure 16: Our sequence of ResNet blocks in the spatial feature and refinement networks.

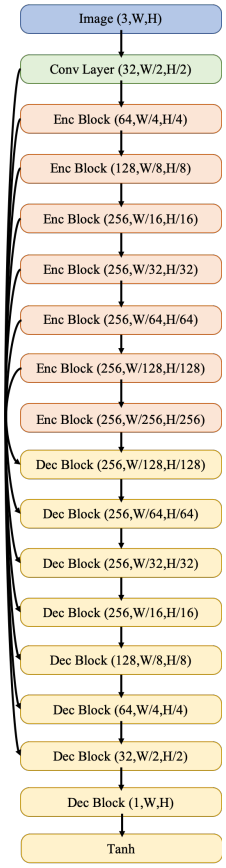


Figure 17: Depth regressor network. An *Enc Block* consists of a sequence of Leaky ReLU, convolution (stride 2, padding 1, kernel size 4), and batch normalisation layers. A *Dec Block* consists of a sequence of ReLU, 2x bilinear upsampling, convolution (stride 1, padding 1, kernel size 3), and batch normalisation layers (except for the final layer, which has no batch normalisation layer).

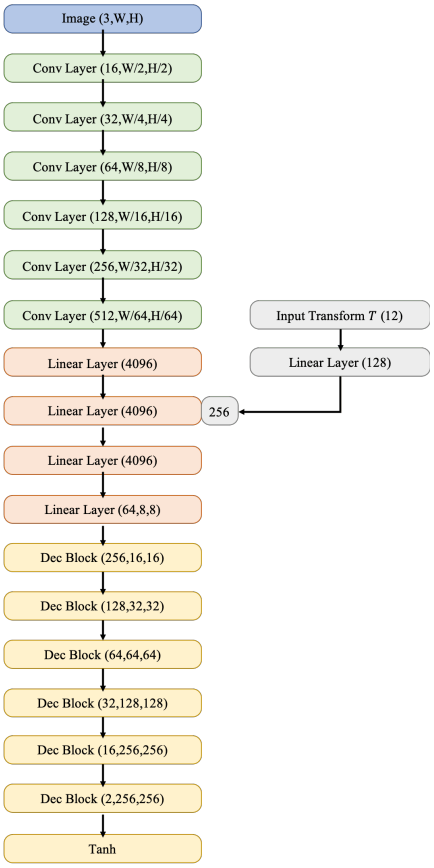
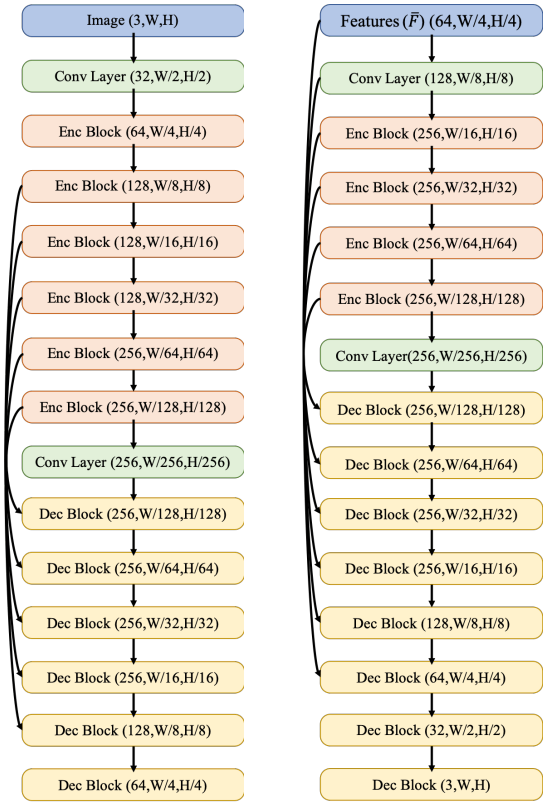


Figure 18: An overview of the image to image network. A *Conv Layer* consists of a sequence of a convolutional layer (stride 2, padding 1, filter size 3), ReLU, and batch normalisation layer. A *Linear Layer* consists of a sequence of a linear layer, ReLU, and batch normalisation layer. A *Dec block* consists of a sequence of a convolutional layer (stride 1, padding 1, filter size 3), ReLU, batch normalisation layer and upsample layer (except for the last, which consists of simply a convolutional layer).



(a) Encoder network. (b) Decoder network.

Figure 19: The encoder and decoder network for the UNet style encoder/decoder setup. An *Enc block* is a sequence of a LeakyReLU, convolutional layer (stride 2, padding 1, kernel size 4) and batch normalisation layer. A *Dec block* is a sequence of ReLU, bilinear upsampling layer, convolutional layer (stride 1, padding 1, kernel size 3), and batch normalisation layer (except for the last layer which has no batch normalisation).

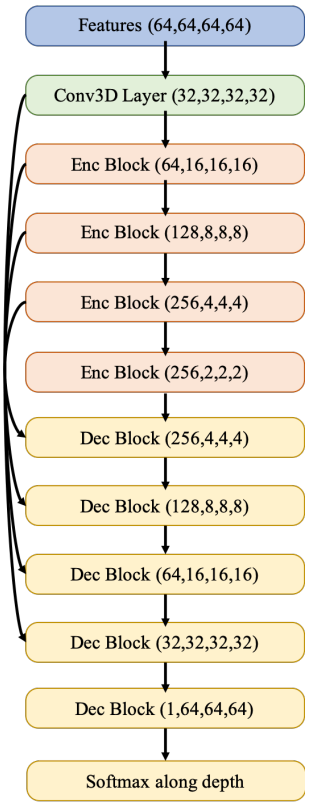
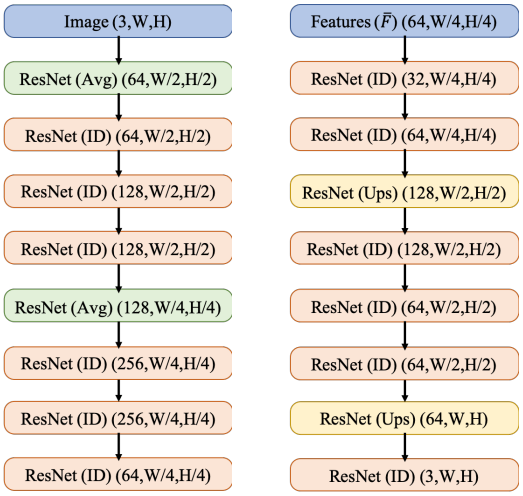


Figure 20: The 3D UNet for predicting the occupancy of voxels. An *Enc block* consists of a sequence of a LeakyReLU, convolutional layer (stride 2, padding 1, kernel size 4) and batch normalisation layer. A *Dec block* consists of a sequence of ReLU, bilinear upsampling layer, convolutional layer (stride 1, padding 1, kernel size 3), and batch normalisation layer (except for the last layer which has no batch normalisation).



(a) Encoder network. (b) Decoder network.

Figure 21: The spatial feature and refinement networks for the ResNet style setup in the Vox w/ ours baseline.

D. Additional information about datasets

Matterport3D. For Matterport, the minimum depth is 0.1 and the maximum depth 10.

RealEstate10K. For RealEstate10K, the minimum depth is 1 and the maximum depth is 100.

KITTI. For KITTI, the minimum depth is 1 and the maximum depth is 50.

E. A description of other setups we tried

Model setup

- We experimented with using a UNet architecture instead of a sequence of ResNet blocks for the spatial feature network and refinement network. This led to much worse results and was more challenging to train.

Differentiable renderer setup

- Other settings for the differentiable renderer: We tried a larger radius, $r = 8$, but this both takes longer to train and gives worse results.
- Other settings for the accumulation function: We tried using a weighted sum with and without normalisation for the accumulation step. These led to similar results, but without normalisation had noisier training characteristics. The implementation of these different accumulation setups is available in the online code.

Chapter 10

Conclusions and Future Work

In this thesis we have discussed a number of methods for using neural networks to predict 3D structure from one or more images. Here we discuss the impact of this work in Section 10.1, possible extensions in Section 10.2, and some final thoughts on the future directions of learning 3D shape from images in Section 10.3.

10.1 Conclusions and Achievements

Conditioning correspondences on both images. In chapter 3, we introduced a new approach for predicting matches between images by conditioning the descriptors to be matched on *both* images. An additional important contribution was to introduce a distinctiveness score which is used to select the best matches. Using this approach, we achieved state of the art or comparable results on a number of datasets and tasks: in particular, local matching, camera localisation, and 3D reconstruction. While not yet published, we hope that this work will encourage future researchers to think how neural networks can be better leveraged to obtain high quality robust matches between images.

Self-supervised learning of photometrically invariant 3D representations. In chapter 4, we introduced a dataset for predicting 3D objects from images. This dataset differs from previous synthetic datasets, as it includes a variety of objects that do *not* come from the same object class; the objects within the dataset have highly accurate textures, as they are based on real sculptures. Using this dataset, we learned an embedding that encodes full 3D shape information by training with silhouette and depth information. Finally, we demonstrated that embeddings from multiple views of the same object can be combined in order to improve the 3D prediction *even* if

the views are not photometrically consistent. This is not possible with traditional approaches.

Self-supervised learning of depth from geometric correspondences. We investigated and demonstrated in chapter 5 how to use correspondences obtained from multi-view geometry in order to train a 3D depth predictor. We first demonstrated this on synthetic data, obtaining high quality results. We then demonstrated this approach on a dataset of sculptures. Current work that uses self-supervision to learn depth from pairs of images uses a photometric loss (meaning that one image rendered using the predicted depth should match, in RGB space, the other image). It would not have been possible to train a network with a photometric loss to predict depth on our sculpture dataset, as the dataset consists of images of a sculpture with challenging variations in illumination, context, and viewpoint. Using a self-supervised loss based on geometric correspondences meant that we could apply our model to this challenging dataset.

In both chapters 4 and 5, we focused on models that can be applied to realistic data. This focus was appreciated by the computer vision community. SilNet (chapter 4) received best poster at the *British Machine Vision Conference* and was extended as a journal article in the *International Journal of Computer Vision*. LiftNet (chapter 5) was selected as an oral presentation at the *Geometry Meets Deep Learning Workshop* at the *European Conference on Computer Vision*.

Self-supervised learning of class embeddings. In chapters 6 and 7, we introduced a self-supervised method for learning a representation of a specific class. Given an image of an instance of an object class, the corresponding learned embedding encodes the pose and structure of the object within the image. We believe we are the first to formalise the self-supervised task in this manner. To learn this representation, we made use of vast amounts of video data and a simple reconstruction task: given two images of the same video, the task is to transform one image into the position of the other. To apply such an approach on more challenging data, such as human pose, which exhibits articulated motion, we generalised our approach using a hierarchical loss. We achieved state of the art results for landmark estimation, approaching the performance of *supervised* methods. Our embedding was also found to be useful for deep-fake detection by Agarwal et al. [2020], as it encodes pose and expression.

Novel view synthesis of faces. In chapter 8, we introduced a method for manipulating a face with another face or other modalities such as pose, or audio. An important contribution was that we demonstrated how a self-supervised approach, trained on large amounts of video data, could be used to manipulate a face with another face *without* any further training. We were the first to solve this task in an entirely self-supervised manner. This was done by enforcing that the model learned a disentangled representation such that one portion of the network learned a canonical face representation and another portion learned how to manipulate this representation into a desired pose and expression. A second contribution was that we showed that the learned representation implicitly encoded higher level information, such as pose or expression, so we could modify this embedding directly to manipulate a face with another modality.

These results demonstrated that a morphable model is *not* needed to manipulate faces and achieve high quality results; we demonstrated that a powerful model for animating still images can be learned in an *entirely* data-driven, self-supervised manner. This greatly simplifies the problem, allowing similar ideas to be applied to more object class.

The X2Face code has been starred over 150 times on GitHub and forked over 30 times [Wiles and Koepke, 2020], demonstrating its value to the wider community.

Novel view synthesis of scenes. In chapter 9, we introduced a method for synthesising new images of a scene given a single image and a desired change in viewpoint. To do this, we represented the 3D structure of the scene as a point cloud of features and introduced a differentiable point cloud renderer to render the point cloud into a new viewpoint. A refinement network was then used to fill in missing regions. The whole model (the depth predictor and refinement module) was trained end to end on real world data. To the best of our knowledge, this is the first work to perform view synthesis in such a manner on real world data and achieve high quality results.

While only a few months old, the code already has over 300 stars on GitHub and 30 forks [Wiles et al., 2020]. The code for manipulating and rendering our point cloud has been included in the open source PyTorch3D library by Ravi et al. [2020] for other researchers.

10.2 Possible Extensions

Conditioning correspondences on both images. We demonstrated in chapter 3 that conditioning descriptors on both images can lead to performance gains when obtaining correspondences. We believe that this is only the first step in investigating how best to extract correspondences for two images by conditioning the descriptors on *both* images. Better architectures should be investigated to further improve the accuracy of the predicted correspondences.

We also note that another future direction is how best to use our approach in a 3D reconstruction pipeline. Current 3D reconstruction pipelines [Schönberger and Frahm, 2016, Schönberger et al., 2016] are based on a *detect and describe* approach. Our approach finds dense correspondences, but the same keypoints may not be found across multiple images, affecting the track length and hurting the performance. In a 3D reconstruction pipeline, points must be seen across multiple different images (the track length) in order to triangulate the points and retrieve the corresponding camera information; if a point is seen only in two images, this does not give a strong enough constraint to perform 3D reconstruction. Instead of using the standard approach, we could use a hierarchy of reference views against which other images are registered. This should ensure that the same keypoints are considered and matched across images to improve the track length.

Self-supervised learning of 3D shape from images. The work in chapters 4 and 5 focused on obtaining 3D information for challenging realistic data, e.g. synthetic sculptures or real images of sculptures. A limitation of the work in chapter 5 was the assumption of an affine camera model. This works well for objects far enough from the camera; however, an obvious extension would be to apply this work to perspective cameras.

Further work should consider how the current, standard photometric loss for training 3D models in a self-supervised manner could be improved using the additional supervision constraints investigated in these chapters, such as a silhouette or correspondence loss. This would allow models to be trained in contexts where a photometric loss does not apply: for example, images from the same video but under large changes (for example, images further apart in the video) and images of the same scene taken with large illumination or context changes.

Self-supervised learning of class embeddings. In chapters 6 and 7, we introduced a framework for obtaining class based embeddings from images. There are two main directions for future work: (i) how to improve the model and (ii) further uses of the embedding. The main limitations of the current model are that it cannot ignore background regions if the background dominates the images and it cannot model parts of the object that are visible in some frames and not in others. The second direction is exploring further uses of the embedding. We demonstrated that our embedding can be used to predict downstream 2D information; Agarwal et al. [2020] has demonstrated its use for detecting fakes. However, there are two further obvious uses: using the embedding as a pretraining step in a model that requires understanding class specific properties (for example, emotion or gaze estimation) and using the embedding to explicitly predict full 3D structure.

Novel view synthesis of faces. In chapter 8, we introduced a self-supervised approach for generating new images of a face conditioned on another face, pose information or an audio signal. Most of the limitations of our current pipeline arise from the use a bilinear sampler in order to transform the images. This limits the quality of the learned images and means we cannot hallucinate unseen regions. There are two obvious directions: first, how to leverage auxiliary supervision (in the form of landmarks or a discriminator) to improve the results and remove these limitations and second, how to apply our approach to other domains. A number of works have improved on our work in just this manner: by using auxiliary supervision or by applying our ideas to new object classes (such as humans) [Siarohin et al., 2019, Zakharov et al., 2019].

Novel view synthesis of scenes. While we achieve impressive results for synthesising new views from real world images in chapter 9, there are a variety of interesting directions to explore. First, the model can only hallucinate images with a change in viewpoint within the distribution that the model was trained with. It would be interesting to be able to generalise beyond this distribution. One possible solution is to use an auto-regressive model: one could use a generated image to generate further images.

A second interesting direction is how to combine multiple input images at different viewpoints when generating new scenes. This would require rethinking how to

represent and manipulate the 3D representation in order to make it consistent across multiple generated images.

A final, challenging direction is how to handle non rigid, moving objects within a scene. In this work, we assume the scene is static. However, it would be interesting to be able to hallucinate the motion of people or animals, or the effects of wind or other external forces.

10.3 Open Questions in Learning Shape

In this thesis we have put forward a variety of successful approaches to overcome limitations of current approaches (both traditional and modern). However, this is still an open and exciting research area with a wide array of interesting, open questions and research directions. Here we discuss a few such relevant questions.

How to predict 3D shape from a single image? This is perhaps the holy grail of questions: given *any* image how can we learn a model to predict the underlying 3D structure (at least as well as a human can, considering that there may be ambiguities). It is not clear what the solution is. Recent years have seen the explosion of high quality large datasets of depth maps from realistic images, implying that perhaps the solution is simply more, and better, data collection. But will this prove intractable, as has seemed the case with object detection, where the generalisation gap due to dataset bias between a sample dataset (such as ImageNet by Deng et al. [2009]) and random images is well documented [Torralba and Efros, 2011]? Perhaps, the solution is instead to improve self-supervised 3D estimation on random videos downloaded from YouTube, but this comes with additional challenges of modelling blur, camera artefacts, non rigid objects, and further difficulties.

How to represent 3D shape? One of the most fundamental questions is how to represent 3D shape. Purists may desire to encode no explicit 3D structure and simply use an embedding in order to model all natural scenes. This seems a very challenging proposition and does not make use of known geometric principles. As a result, other approaches propose representations that explicitly encode 3D structure (for instance, in the form of voxels or point clouds or meshes). The best representation is not clear and presumably depends on the task. If we want to perform view synthesis from a single image, then multiplane images [Flynn et al., 2019, Srinivasan et al., 2019]

have been shown to be effective; however, for medical imaging, it is unlikely such a representation is useful.

How to leverage auxiliary data? Currently, 3D prediction and corresponding applications often exist in a sphere independent of other computer vision areas (such as semantic segmentation, instance segmentation or audio modelling). However, these semantic properties, possible scene interactions (for instance, what objects can be picked up or put down) and audio properties are important for many applications. For example, in a real estate application, we may want to move objects around, add in valid possible objects, and perform further actions that require interacting with the 3D environment. This is a nascent area, which has started to see some interesting work [Tulsiani et al., 2018a, Vineet et al., 2015, Wald et al., 2020], but it will presumably grow in scope, as 3D estimation improves and further applications become possible.

How to handle generalisation? As models that predict 3D structure improve, they will become pivotal components of a variety of applications ranging from autonomous driving to robotics. In these conditions, there may be severe consequences if the depth prediction is wrong (for instance, a vehicle crashing or a robot damaging itself, hurting its lifetime). Handling novel conditions requires a model to be robust to new conditions and environments. It also requires being able to determine when the model will work and how it will fail in order to give users confidence in the system.

What are the ethical implications? While not directly related to actually predicting 3D structure, there has been a necessary focus on the ethical problems that may arise with current machine learning methods. When manipulating faces, or another object class, the implications are obvious: such a model can be used to make a person seem to have said or done something fake. The same is true for manipulating scenes, as new objects could be added into a photograph after the event. While it is not the focus of our work to handle these ethical implications, an interesting (and important) direction is to be able to verify whether an image is genuine or has been manipulated. There has indeed been a set of works [Agarwal et al., 2020, Huh et al., 2018, Wang et al., 2020] that investigate how to detect fakes, and this area will presumably grow, as the quality (and thereby danger) of faked images increases.

Bibliography

Sameer Agarwal, Noah Snavely, Ian Simon, Steve Seitz, and Richard Szeliski. Building Rome in a day. In *Proceedings of the International Conference on Computer Vision*, 2009.

Shruti Agarwal, Tarek El-Gaaly, Hany Farid, and Ser-Nam Lim. Detecting deep-fake videos from appearance and behavior. *arXiv preprint arXiv:2004.14491*, 2020.

Kara-Ali Aliev, Dmitry Ulyanov, and Victor Lempitsky. Neural point-based graphics. *arXiv preprint arXiv:1906.08240*, 2019.

Hadar Averbuch-Elor, Daniel Cohen-Or, Johannes Kopf, and Michael F. Cohen. Bringing portraits to life. *ACM Transactions on Graphics (TOG)*, 2017.

Vassileios Balntas, Edgar Riba, Daniel Ponsa, and Krystian Mikolajczyk. Learning local feature descriptors with triplets and shallow convolutional neural networks. In *Proceedings of the British Machine Vision Conference*, 2016.

Jonathan T. Barron and Jitendra Malik. High-frequency shape and albedo from shading using natural image statistics. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2011.

Jonathan T. Barron and Jitendra Malik. Shape, albedo, and illumination from a single image of an unknown object. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, June 2012.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *Proceedings of the European Conference on Computer Vision*, 2006.

Andrew Blake and Constantinos Marinos. Shape from texture: estimation, isotropy and moments. *Artificial Intelligence*, 45(3):323–380, 1990.

Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3D faces. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 1999.

Volker Blanz and Thomas Vetter. Face recognition based on fitting a 3D morphable model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(9):1063–1074, 2003.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J Black. Keep it SMPL: Automatic estimation of 3d human pose and shape from a single image. In *Proceedings of the European Conference on Computer Vision*, 2016.

James Booth, Anastasios Roussos, Stefanos Zafeiriou, Allan Ponniah, and David Dunaway. A 3D morphable model learnt from 10,000 faces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Eric Brachmann, Alexander Krull, Sebastian Nowozin, Jamie Shotton, Frank Michel, Stefan Gumhold, and Carsten Rother. DSAC-Differentiable RANSAC for camera localization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale GAN training for high fidelity natural image synthesis. In *Proceedings of the International Conference on Learning Representations*, 2019.

Matthew Brown, Richard Szeliski, and Simon Winder. Multi-image matching using multi-scale oriented patches. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary robust independent elementary features. In *Proceedings of the European Conference on Computer Vision*, 2010.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. ShapeNet: An information-rich 3D model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Gaurav Chaurasia, Sylvain Duchene, Olga Sorkine-Hornung, and George Drettakis. Depth synthesis and local warps for plausible image-based navigation. *ACM Transactions on Graphics (TOG)*, 2013.

Tao Chen, Zhe Zhu, Ariel Shamir, Shi-Min Hu, and Daniel Cohen-Or. 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, 32(6):1–10, 2013.

Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *Advances in Neural Information Processing Systems*, 2016a.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in Neural Information Processing Systems*, 2016b.

Xu Chen, Jie Song, and Otmar Hilliges. Monocular neural image based rendering with continuous view control. *Proceedings of the International Conference on Computer Vision*, 2019.

Inchang Choi, Orazio Gallo, Alejandro Troccoli, Min H Kim, and Jan Kautz. Extreme view synthesis. In *Proceedings of the International Conference on Computer Vision*, 2019.

Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3D object reconstruction. In *Proceedings of the European Conference on Computer Vision*, 2016.

Joon Son Chung, Amir Jamaludin, and Andrew Zisserman. You said that? In *Proceedings of the British Machine Vision Conference*, 2017.

Joon Son Chung, Arsha Nagrani, and Andrew Zisserman. VoxCeleb2: Deep speaker recognition. In *INTERSPEECH*, 2018.

David Crandall, Andrew Owens, Noah Snavely, and Daniel Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

Kevin Dale, Kalyan Sunkavalli, Micah K Johnson, Daniel Vlasic, Wojciech Matusik, and Hanspeter Pfister. Video face replacement. *ACM Transactions on Graphics (TOG)*, 2011.

Paul Debevec, Yizhou Yu, and George Borshukov. Efficient view-dependent image-based rendering with projective texture-mapping. *Rendering Techniques*, 1998.

Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image- based approach. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, pages 11–20, 1996.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *CVPR workshops*, 2018.

Hui Ding, Kumar Sricharan, and Rama Chellappa. ExprGAN: Facial expression editing with controllable expression intensity. In *Proc. AAAI*, 2018.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the International Conference on Computer Vision*, 2015.

Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A trainable cnn for joint description and detection of local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Advances in Neural Information Processing Systems*, 2014.

Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3D object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Olivier Faugeras and Q.-T. Luong. *The Geometry of Multiple Images*. MIT Press, 2001.

M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. Assoc. Comp. Mach.*, 24(6):381–395, 1981.

A. W. Fitzgibbon, Y. Wexler, and A. Zisserman. Image-based rendering using image-based priors. *International Journal of Computer Vision*, 63(2):141–151, 2005.

John Flynn, Ivan Neulander, James Philbin, and Noah Snavely. Deepstereo: Learning to predict new views from the world's imagery. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

John Flynn, Michael Broxton, Paul Debevec, Matthew DuVall, Graham Fyffe, Ryan Overbeck, Noah Snavely, and Richard Tucker. Deepview: View synthesis with learned gradient descent. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Wolfgang Förstner and Eberhard Gülch. A fast operator for detection and precise location of distinct points, corners and center of circular features. In *Proc. of ISPRS Intercommission Conference on Fast Processing of Photogrammetric Data, Interlaken, Switzerland*, pages 281–305, June 1987.

Yasutaka Furukawa and Jean Ponce. Accurate, dense, and robust multiview stereopsis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(8), 2009.

Jonas Gårding. Shape from texture for smooth curved surfaces in perspective projection. *Journal of Mathematical Imaging and Vision*, 2(4):327–350, 1992.

Ravi Garg, Vijay Kumar BG, Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *Proceedings of the European Conference on Computer Vision*, 2016.

Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research (IJRR)*, 2013.

Rohit Girdhar, David F Fouhey, Mikel Rodriguez, and Abhinav Gupta. Learning a predictable and generative vector representation for objects. In *Proceedings of the European Conference on Computer Vision*, 2016.

Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh r-cnn. *Proceedings of the International Conference on Computer Vision*, 2019.

Clément Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Ariel Gordon, Hanhan Li, Rico Jonschkowski, and Anelia Angelova. Depth from videos in the wild: Unsupervised monocular depth learning from unknown cameras. In *Proceedings of the International Conference on Computer Vision*, 2019.

Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. Atlasnet: A papier-mâché approach to learning 3d surface generation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Christian Häne, Shubham Tulsiani, and Jitendra Malik. Hierarchical surface prediction for 3D object reconstruction. In *2017 International Conference on 3D Vision (3DV)*, 2017.

Christopher G. Harris and Mike Stephens. A combined corner and edge detector. In *Proceedings of the 4th Alvey Vision Conference, Manchester*, pages 147–151, 1988.

Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521623049, 2000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Transactions on Image Processing*, 2018.

Peter Hedman, Julien Philip, True Price, Jan-Michael Frahm, George Drettakis, and Gabriel Brostow. Deep blending for free-viewpoint image-based rendering. *ACM Transactions on Graphics (TOG)*, 2018.

Olivier J Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, SM Eslami, and Aaron van den Oord. Data-efficient image recognition with contrastive predictive coding. *arXiv preprint arXiv:1905.09272*, 2019.

G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786), 2006.

Po-Han Huang, Kevin Matzen, Johannes Kopf, Narendra Ahuja, and Jia-Bin Huang. DeepMVS: Learning multi-view stereopsis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Minyoung Huh, Andrew Liu, Andrew Owens, and Alexei A Efros. Fighting fake news: Image splice detection via learned self-consistency. In *Proceedings of the European Conference on Computer Vision*, 2018.

Eldar Insafutdinov and Alexey Dosovitskiy. Unsupervised learning of shape and pose with differentiable point clouds. In *Advances in Neural Information Processing Systems*, 2018.

Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. SurfaceNet: An end-to-end 3D neural network for multiview stereopsis. In *Proceedings of the International Conference on Computer Vision*, 2017.

Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018a.

Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018b.

Abhishek Kar, Shubham Tulsiani, Joao Carreira, and Jitendra Malik. Category-specific object reconstruction from a single image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *Advances in Neural Information Processing Systems*, 2017.

Yan Ke and Rahul Sukthankar. PCA-SIFT: A more distinctive representation for local image descriptors. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Washington, DC*, June 2004.

Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *Proceedings of the International Conference on Computer Vision*, 2017.

Sameh Khamis, Jonathan Taylor, Jamie Shotton, Cem Keskin, Shahram Izadi, and Andrew Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

Natasha Kholgade, Tomas Simon, Alexei Efros, and Yaser Sheikh. 3D object manipulation in a single photograph using stock 3d models. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014.

Tejas Khot, Shubham Agrawal, Shubham Tulsiani, Christoph Mertz, Simon Lucey, and Martial Hebert. Learning unsupervised multi-view stereopsis via robust photometric consistency. *arXiv preprint arXiv:1905.02706*, 2019.

Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *ACM international symposium on mixed and augmented reality*, 2007.

Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *International Symposium on Mixed and Augmented Reality*, 2009.

Reinhard Koch, Marc Pollefeys, and Luc Van Gool. Multi viewpoint stereo from uncalibrated video sequences. In *Proceedings of the European Conference on Computer Vision*, 1998.

Johannes Kopf, Fabian Langguth, Daniel Scharstein, Richard Szeliski, and Michael Goesele. Image-based rendering in the gradient domain. *ACM Transactions on Graphics (TOG)*, 2013.

Alex Krizhevsky, Ilya Sutskever, and Geoffery E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1106–1114, 2012.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. In *Advances in Neural Information Processing Systems*, 2015.

Iro Laina, Christian Rupprecht, Vasileios Belagiannis, Federico Tombari, and Nassir Navab. Deeper depth prediction with fully convolutional residual networks. In *2016 Fourth international conference on 3D vision (3DV)*, 2016.

Aldo Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, February 1994.

Stefan Leutenegger, Margarita Chli, and Roland Siegwart. BRISK: Binary robust invariant scalable keypoints. In *Proceedings of the International Conference on Computer Vision*, pages 2548–2555, 2011.

Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Zhengqi Li, Tali Dekel, Forrester Cole, Richard Tucker, Noah Snavely, Ce Liu, and William T Freeman. Learning the depths of moving people by watching frozen people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. *Proceedings of the International Conference on Computer Vision*, 2019a.

Shichen Liu, Shunsuke Saito, Weikai Chen, and Hao Li. Learning to infer implicit surfaces without 3D supervision. In *Advances in Neural Information Processing Systems*, 2019b.

Stephen Lombardi, Tomas Simon, Jason Saragih, Gabriel Schwartz, Andreas Lehrmann, and Yaser Sheikh. Neural volumes: Learning dynamic renderable volumes from images. *ACM Transactions on Graphics (TOG)*, 2019.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, 2015.

David Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. of the 7th International Joint Conference on Artificial Intelligence*, pages 674–679, 1981. URL citeseer.nj.nec.com/lucas81optical.html.

Jitendra Malik and Ruth Rosenholtz. Computing local surface orientation and shape from texture for curved surfaces. *International Journal of Computer Vision*, 2(23):143–168, 1997.

Ricardo Martin-Brualla, David Gallup, and Steven M Seitz. Time-lapse mining from internet photos. *ACM Transactions on Graphics (TOG)*, 2015a.

Ricardo Martin-Brualla, David Gallup, and Steven M Seitz. 3D time-lapse reconstruction from internet photos. In *Proceedings of the International Conference on Computer Vision*, 2015b.

Ricardo Martin-Brualla, Rohit Pandey, Shuoran Yang, Pavel Pidlypenskyi, Jonathan Taylor, Julien Valentin, Sameh Khamis, Philip Davidson, Anastasia Tkach, Peter Lincoln, et al. Lookingood: enhancing performance capture with real-time neural re-rendering. *ACM Transactions on Graphics (TOG)*, 2018.

Kevin Matzen and Noah Snavely. Scene chronology. In *Proceedings of the European Conference on Computer Vision*, 2014.

Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy Networks: Learning 3D reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Moustafa Meshry, Dan B Goldman, Sameh Khamis, Hugues Hoppe, Rohit Pandey, Noah Snavely, and Ricardo Martin-Brualla. Neural rerendering in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Mateusz Michalkiewicz, Jhony K Pontes, Dominic Jack, Mahsa Baktashmotlagh, and Anders Eriksson. Deep level sets: Implicit surface representations for 3d shape inference. *arXiv preprint arXiv:1901.06802*, 2019.

Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International Journal of Computer Vision*, 1(60):63–86, 2004.

Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1/2):43–72, 2005.

Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. *arXiv preprint arXiv:2003.08934*, 2020.

Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Thu Nguyen-Phuoc, Chuan Li, Lucas Theis, Christian Richardt, and Yong-Liang Yang. HoloGAN: Unsupervised learning of 3d representations from natural images. In *Proceedings of the International Conference on Computer Vision*, 2019.

Thu H Nguyen-Phuoc, Chuan Li, Stephen Balaban, and Yongliang Yang. RenderNet: A deep convolutional network for differentiable rendering from 3D shapes. In *Advances in Neural Information Processing Systems*, 2018.

Simon Niklaus, Long Mai, Jimei Yang, and Feng Liu. 3D Ken Burns effect from a single image. *ACM Transactions on Graphics (TOG)*, 2019.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *Proceedings of the European Conference on Computer Vision*, pages 69–84. Springer, 2016.

David Novotny, Ben Graham, and Jeremy Reizenstein. PerspectiveNet: A scene-consistent image generator for new view synthesis in real indoor environments. In *Advances in Neural Information Processing Systems*, 2019.

Kyle Olszewski, Zimo Li, Chao Yang, Yi Zhou, Ronald Yu, Zeng Huang, Sitao Xiang, Shunsuke Saito, Pushmeet Kohli, and Hao Li. Realistic dynamic facial textures from a single image using GANs. In *Proceedings of the International Conference on Computer Vision*, 2017.

Kyle Olszewski, Sergey Tulyakov, Oliver Woodford, Hao Li, and Linjie Luo. Transformable bottleneck networks. In *Proceedings of the International Conference on Computer Vision*, 2019.

Yuki Ono, Eduard Trulls, Pascal Fua, and Kwang Moo Yi. LF-Net: learning local features from images. 2018.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

Eunbyung Park, Jimei Yang, Ersin Yumer, Duygu Ceylan, and Alexander C Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2536–2544, 2016.

Georgios Pavlakos, Luyang Zhu, Xiaowei Zhou, and Kostas Daniilidis. Learning to estimate 3D human pose and shape from a single color image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Eric Penner and Li Zhang. Soft 3D reconstruction for view synthesis. *ACM Transactions on Graphics (TOG)*, 2017.

M. Pollefeys, R. Koch, and L. Van Gool. Self calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In *Proceedings of the International Conference on Computer Vision*, 1998.

René Ranftl and Vladlen Koltun. Deep fundamental matrix estimation. In *Proceedings of the European Conference on Computer Vision*, 2018.

René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *arXiv:1907.01341*, 2019.

Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. PyTorch3D. https://github.com/facebookresearch/pytorch3d, 2020.

Danilo Jimenez Rezende, SM Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3D structure from images. In *Advances in Neural Information Processing Systems*, 2016.

Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning deep 3D representations at high resolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Sami Romdhani, Volker Blanz, and Thomas Vetter. Face identification by fitting a 3D morphable model using linear shape and texture error functions. In *Proceedings of the European Conference on Computer Vision*, 2002.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary R. Bradski. Orb: An efficient alternative to SIFT or SURF. In *Proceedings of the International Conference on Computer Vision*, pages 2564–2571, 2011.

Dimitrios Samaras, Dimitris Metaxas, Pascal Fua, and Yvan G Leclerc. Variable albedo surface reconstruction from stereo and shape from shading. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2000.

Nikolay Savinov, Akihito Seki, Lubor Ladicky, Torsten Sattler, and Marc Pollefeys. Quad-Networks: unsupervised learning to rank for interest point detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Frederik Schaffalitzky and Andrew Zisserman. Multi-view matching for unordered image sets, or "How do i organize my holiday snaps?". In *Proceedings of the European Conference on Computer Vision*, 2002.

Cordelia Schmid, Roger Mohr, and Christian Bauckhage. Evaluation of interest point detectors. *International Journal of Computer Vision*, 37(2):151–172, 2000.

Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys, and Jan-Michael Frahm. Pixelwise view selection for unstructured multi-view stereo. In *Proceedings of the European Conference on Computer Vision*, 2016.

Meng-Li Shih, Shih-Yang Su, Johannes Kopf, and Jia-Bin Huang. 3D photography using context-aware layered depth inpainting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Daeyun Shin, Zhile Ren, Erik Sudderth, and Charless Fowlkes. 3D scene reconstruction with multi-layer depth and epipolar transformers. In *Proceedings of the International Conference on Computer Vision*, 2019.

Aliaksandr Siarohin, Stephane Lathuiliere, Sergey Tulyakov, Elisa Ricci, and Nicu Sebe. Animating arbitrary objects via deep motion transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *Proceedings of the European Conference on Computer Vision*, 2012.

Edgar Simo-Serra, Eduard Trulls, Luis Ferraz, Iasonas Kokkinos, Pascal Fua, and Francesc Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *Proceedings of the International Conference on Computer Vision*, 2015.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Learning local feature descriptors using convex optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

Vincent Sitzmann, Justus Thies, Felix Heide, Matthias Nießner, Gordon Wetzstein, and Michael Zollhofer. DeepVoxels: Learning persistent 3D feature embeddings. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019a.

Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3D-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019b.

Noah Snavely, Steve Seitz, and Richard Szeliski. Photo tourism: exploring photo collections in 3D. In *Proceedings of the ACM SIGGRAPH Conference on Computer Graphics*, number 3, pages 835–846, 2006.

Pratul P Srinivasan, Tongzhou Wang, Ashwin Sreelal, Ravi Ramamoorthi, and Ren Ng. Learning to synthesize a 4D rgbd light field from a single image. In *Proceedings of the International Conference on Computer Vision*, 2017.

Pratul P Srinivasan, Richard Tucker, Jonathan T Barron, Ravi Ramamoorthi, Ren Ng, and Noah Snavely. Pushing the boundaries of view extrapolation with multiplane images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Shao-Hua Sun, Minyoung Huh, Yuan-Hong Liao, Ning Zhang, and Joseph J Lim. Multi-view to novel view: Synthesizing novel views with self-learned confidence. In *Proceedings of the European Conference on Computer Vision*, 2018.

Supasorn Suwajanakorn, Steven M Seitz, and Ira Kemelmacher-Shlizerman. Synthesizing obama: learning lip sync from audio. *ACM Transactions on Graphics (TOG)*, 36(4):95, 2017.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Multi-view 3D models from single images with a convolutional network. In *Proceedings of the European Conference on Computer Vision*, 2016.

Ayush Tewari, Michael Zollhofer, Hyeongwoo Kim, Pablo Garrido, Florian Bernard, Patrick Perez, and Christian Theobalt. MoFa: Model-based deep convolutional face autoencoder for unsupervised monocular reconstruction. In *Proceedings of the International Conference on Computer Vision*, 2017.

Justus Thies, Michael Zollhöfer, Marc Stamminger, Christian Theobalt, and Matthias Nießner. Face2Face: Real-time face capture and reenactment of RGB videos. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

Justus Thies, Michael Zollhöfer, Christian Theobalt, Marc Stamminger, and Matthias Nießner. Image-guided neural object rendering. *Proceedings of the International Conference on Learning Representations*, 2020.

Antonio Torralba and Alexei A Efros. Unbiased look at dataset bias. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

Luan Tran and Xiaoming Liu. Nonlinear 3D face morphable model. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

W. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment: A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000.

Anh Tuan Tran, Tal Hassner, Iacopo Masi, and Gérard Medioni. Regressing robust and discriminative 3D morphable models with a very deep neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017a.

Shubham Tulsiani, Tinghui Zhou, Alexei A Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017b.

Shubham Tulsiani, Saurabh Gupta, David F Fouhey, Alexei A Efros, and Jitendra Malik. Factoring shape, pose, and layout from the 2D image of a 3D scene. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018a.

Shubham Tulsiani, Richard Tucker, and Noah Snavely. Layer-structured 3D scene inference via view synthesis. In *Proceedings of the European Conference on Computer Vision*, 2018b.

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 2010.

Vibhav Vineet, Ondrej Miksik, Morten Lidegaard, Matthias Nießner, Stuart Golodetz, Victor A Prisacariu, Olaf Kähler, David W Murray, Shahram Izadi, Patrick Pérez, and Philip H. S. Torr. Incremental dense semantic stereo fusion for large-scale semantic scene reconstruction. In *Proceedings of the International Conference on Robotics and Automation*, 2015.

Daniel Vlasic, Matthew Brand, Hanspeter Pfister, and Jovan Popović. Face transfer with multilinear models. *ACM Transactions on Graphics (TOG)*, 2005.

Johanna Wald, Helisa Dhamo, Nassir Navab, and Federico Tombari. Learning 3D semantic scene graphs from 3D indoor reconstructions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *Proceedings of the European Conference on Computer Vision*, 2018.

Sheng-Yu Wang, Oliver Wang, Richard Zhang, Andrew Owens, and Alexei A Efros. CNN-generated images are surprisingly easy to spot...for now. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

Olivia Wiles and A. Sophia Koepke. X2Face Github Code. https://github.com/oawiles/X2Face, June 2020.

Olivia Wiles, Georgia Gkioxari, and Justin Johnson. SynSin GitHub Code. https://github.com/facebookresearch/synsin, June 2020.

Andrew P. Witkin. Recovering surface shape and orientation from texture. *Artificial Intelligence*, 17(1–3):17–45, August 1981.

Daniel E. Worrall, Stephan J. Garbin, Daniyar Turmukhambetov, and Gabriel J. Brostow. Interpretable transformations with encoder-decoder networks. In *Proceedings of the International Conference on Computer Vision*, 2017.

Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. MarrNet: 3D shape reconstruction via 2.5D sketches. In *Advances in Neural Information Processing Systems*, 2017.

Zexiang Xu, Sai Bi, Kalyan Sunkavalli, Sunil Hadap, Hao Su, and Ravi Ramamoorthi. Deep view synthesis from sparse photometric images. *ACM Transactions on Graphics (TOG)*, 2019.

Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, 2016.

Yao Yao, Zixin Luo, Shiwei Li, Tianwei Shen, Tian Fang, and Long Quan. Recurrent MVSNet for high-resolution multi-view stereo depth inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned invariant feature transform. In *Proceedings of the European Conference on Computer Vision*, 2016.

Egor Zakharov, Aliaksandra Shysheya, Egor Burkov, and Victor Lempitsky. Few-shot adversarial learning of realistic neural talking head models. In *Proceedings of the International Conference on Computer Vision*, 2019.

Linguang Zhang and Szymon Rusinkiewicz. Learning to detect features in texture images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

Richard Zhang, Phillip Isola, and Alexei A Efros. Split-brain autoencoders: Unsupervised learning by cross-channel prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Ruo Zhang, Ping-Sing Tsai, James Edwin Cryer, and Mubarak Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In *Proceedings of the European Conference on Computer Vision*, 2016.

Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *ACM Transactions on Graphics (TOG)*, 2018.

C Lawrence Zitnick, Sing Bing Kang, Matthew Uyttendaele, Simon Winder, and Richard Szeliski. High-quality video view interpolation using a layered representation. *ACM transactions on graphics (TOG)*, 2004.

Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

Silvia Zuffi, Angjoo Kanazawa, Tanja Berger-Wolf, and Michael J. Black. Three-D safari: Learning to estimate zebra pose, shape, and texture from images 'in the wild'. In *Proceedings of the International Conference on Computer Vision*, 2019.