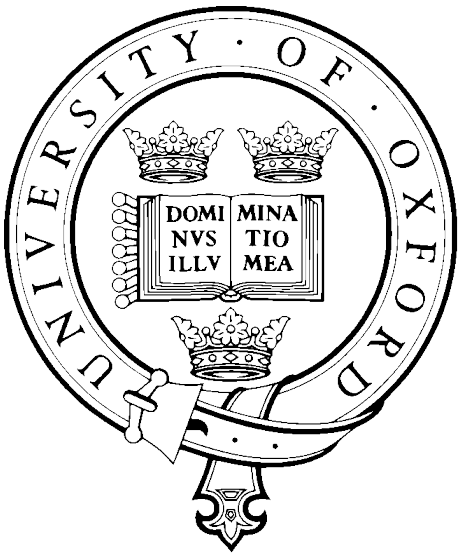
Long Range Monocular SLAM

Duncan Peter Frost

Balliol College



Robotics Research Group

Department of Engineering Science

University of Oxford

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

Abstract

This thesis explores approaches to two problems in the frame-rate computation of *a priori* unknown 3D scene structure and camera pose using a single camera, or monocular simultaneous localisation and mapping. The thesis reflects two trends in vision in general and structure from motion in particular: (i) the move from directly recovered and towards learnt geometry; and (ii) the sparsification of otherwise dense direct methods.

The first contributions mitigate scale drift. Beyond the inevitable accumulation of random error, monocular SLAM accumulates error via the depth/speed scaling ambiguity. Three solutions are investigated. The first detects objects of known class and size using fixed descriptors, and incorporates their measurements in the 3D map. Experiments using databases with ground truth show that metric accuracy can be restored over kilometre distances; and similar gains are made using a hand-held camera. Our second method avoids explicit feature choice, instead employing a deep convolutional neural network to yield depth priors. Relative depths are learnt well, but absolute depths less so, and recourse to database-wide scaling is investigated. The third approach uses a novel trained network to infer speed from imagery.

The second part of the thesis develops sparsified direct methods for monocular SLAM. The first contribution is a novel camera tracker operating directly using affine image warping, but on patches around sparse corners. Camera pose is recovered with an accuracy at least equal to the state of the art, while requiring only half the computational time. The second introduces a least squares adjustment to sparsified direct map refinement, again using patches from sparse corners. The accuracy of its 3D structure estimation is compared with that from the widely used method of depth filtering. It is found empirically that the new method's accuracy is often higher than that of its filtering counterpart, but that the method is more troubled by occlusion.

Acknowledgements

Firstly, I owe a great debt of gratitude to my supervisor David Murray for his continued support over the years.

I would also like to thank my family and Anna, without whom none of this would have been possible.

Finally, I would like to thank past and present members of the Active Vision Laboratory and Visual Geometry Group. Our conversations have been both insightful and essential in maintaining my sanity.

Contents

Chapter 1

Introduction

This chapter motivates for the research in this thesis, which pertains to the topics of scale drift and sparsified direct methods in monocular SLAM. An overview of the thesis and its contributions is also presented.

1.1 Objectives

This thesis reports on studies of two computational issues in the frame-rate recovery of unknown 3D scene structure and 6 degree of freedom camera pose using a single camera.

The broad field of structure from motion (SfM) and simultaneous localisation and mapping (SLAM) is an extremely well-studied one. SfM was first researched intensively in the 1980s with initial emphases on (i) simple interpretations of optical flow fields and (ii) optimal reconstruction of minimalist (n-point) structure (*e.g.* [1]). Later, as computational power increased, the focus advanced to the recovery of dense structure computed *off-line* in batch mode (*e.g.* [2]). In contrast, simultaneous localisation and mapping has its hinterland and tradition in robotics [3], initially using sonar and laser range finders, with an emphasis on the *on-line* recursive recovery of the scene and sensor position within it. The first monocular visual SLAM algorithm [4] [5] — like the original robotics SLAM formulation [6] and the first stereo algorithm [7, 8] — combined current measurements and prior state vector using an EKF. monoSLAM's wider significance is that it made live scene recovery and camera pose tracking feasible for a wide range of applications using hand-held cameras and a laptop computer, immediately freeing SLAM from traditional mechanised robotics.

But a shortcoming of all single camera methods is their inability to resolve the depth/speed

scaling ambiguity: is the camera moving quickly in a large world, or slowly in a miniature one? So, beyond the inevitable accumulation of random error, monocular SLAM accumulates error, or *scale drift*, through this ambiguity. This provides the objective for the first part of this thesis — *the mitigation of scale drift*.

In the decade or so since the introduction of monoSLAM, progress in single camera reconstruction and tracking has centred on scene (and hence image) representation, strongly influenced by available computational power. One difficulty with SLAM based on the extended Kalman filter (EKF) is its quadratic complexity in the number of scene points, which unreasonably restricts the map size, and a second is that the motion model is too constraining. Parallel tracking and mapping (PTAM) [9] separated the urgent task of frame-to-frame camera tracking from the more measured task of map building. Camera tracking is treated as a problem in visual odometry, and mapping as one of optimal reconstruction from keyframes using bundle adjustment. The manageable number of scene points, and hence extent of the map, increase by two orders of magnitude. With increases in computational power available and the advent of GPGPUs, *dense* mapping became possible. By leveraging its inherent parallelisability, the well-studied problem of using *all* image information to build a dense 3D map [10] could be achieved in real-time. Many works [11, 12, 13] estimated dense maps in real-time, but still required camera poses obtained from sparse feature-based methods. DTAM [14] used a dense representation in both map-building *and* camera localisation. By tracking the camera using raw image information, camera estimates were invariant to motion blur and camera defocus. Mobilefusion [15] similarly reconstructs and tracks using a dense map representation, but is limited to small objects of interest.

But the need for graphics processors runs somewhat counter to the use of single cameras as part of power-efficient package, and the most recent trend is to explore sparsification of otherwise direct dense methods such as LSD-SLAM [16], where depth is only estimated around regions with high image-gradient. This provides the objective for the second part of this thesis — *an exploration of sparsification back to the point level*.

We consider these motivations in some more detail.

1.1.1 Objective I: Scale drift

Due to the purely projective nature of a single camera, three-dimensional structure may only be recovered up to scale in monocular SLAM. While this global scale-ambiguity can be resolved by

Figure 1.1: Richard Wiseman's video *Assumptions* [17] fools viewers into thinking objects are bigger or smaller than they really are by exploiting their prior assumptions about the scale of objects.

assuming the size of the camera's initial translation (*e.g.* [9, 18]), maps estimated using monocular methods still have *seven* degrees of gauge freedom [19]. The degrees of gauge freedom define how the entire estimated map may be transformed, without affecting the values at sensor measurements [20]. While SLAM methods that can measure absolute depth [21, 22, 23, 24, 25, 26] will experience drift in their translation and rotation estimates, the additional degree of gauge freedom in monocular SLAM results in drift in *scale* as well [19]. Unlike translational or rotational drift, scale drift often leads to complete positioning failure as the estimated map eventually becomes infinitesimally large or small.

A cyclopean human observer moving through a particular environment does not suffer from scale drift [27, 28]. Even with one eye closed, humans are able to draw on a wealth of prior information about the size of everyday objects to infer a sense of scale. *Relative size* optical illusions (*e.g.* in Fig. 1.1) make use of this: so strong is the assumption that a teacup image is of a conventional teacup, that the viewer is fooled without question. The research presented in Chapter 2 similarly uses priors on object size to resolve scale. We also borrow (but do not mimick) human traits by employing convolutional neural networks [29, 30] to leverage large amounts of prior information acquired during training to perform inference. We propose using these networks to provide hints of scale or depth in monocular SLAM.

1.1.2 Objective II: Sparse direct mapping

In an attempt to reduce computational cost, single camera SLAM has made widespread use of sparse features extracted from the image for camera tracking and map refinement [4, 5, 9, 31, 18].

These use a *geometric* error term, that penalises the difference between a landmark's predicted and measured projection in the camera frame.

Rather than penalise geometric error, direct methods penalise *photometric* error: the difference in predicted and observed pixel intensities. Rather than using a sparse set of features, the entire image is be used to create dense reconstructions of the environment [12, 14, 32, 33]. Due to the large amount of processing required, GPUs are often used to ensure real-time performance.

Besides having denser reconstructions, direct methods have a number of advantages over feature-based methods, such as invariance to motion blur and camera defocus. Recent works [16, 34] seek to gain these advantages without the high computational costs that come with them. Rather than use a GPU, they achieve real-time performance on a CPU by reducing photometric error evaluation to sparse regions in the image.

Current sparsified direct methods either operate as [35, 34] or very similarly to [16, 36] visual odometry methods, estimating a relative pose between the current tracking frame and a keyframe in the map. Structural estimation often uses a form of filtering [32], where past measurements are used to update a distribution over estimated parameters. These filters often rely on assumptions about the distribution of depth measurements.

For feature-based methods, bundle-adjustment [19] has long been considered more accurate than filtering [37]. So far, no work has investigated whether this is the case for sparsified direct methods.

For this reason, we motivate for the investigation of unexplored aspects of sparsified direct methods: How methods can be made more globally consistent during tracking, by estimating the current pose relative to a larger map rather than a single reference frame; and how filtering might be replaced by a direct bundle adjustment that requires less assumptions about the data and uses larger baselines while still maintaining real-time performance.

1.2 Thesis structure and contributions

Here, the main thesis contributions are discussed. This thesis explores solutions to the problem of scale drift in monocular SLAM and investigates sparsified direct methods.

In Chapter 2, scale drift is corrected by detecting objects of a known class in the video sequence. If the class's size distribution is known and has low-variance, object detections are able to

provide hints about the true scale of the estimated map. Objects are localised from measurements and added to the map alongside sparse landmarks. A novel bundle adjustment is then introduced that jointly optimises landmarks, objects and camera poses. By utilising the approximate size distributions of objects in the optimisation, it is able to produce metrically accurate camera trajectories with no scale drift for video filmed over distances kilometres in length. This work was published in ICRA2016 [38].

While metrically accurate, the method proposed in Chapter 2 relies on frequent observations of the known object-class throughout the video sequence. This proved restrictive in a more general setting. Chapter 3 takes advantage of current deep learning research that infers depth from a single monocular image. The depth maps from video frames are used to place priors on the depths of landmarks in each image. A novel bundle adjustment is introduced that jointly minimises geometric error and a constraint term that penalises landmarks that do not adhere to priors. The method is able to maintain scale drift-free operation on a large outdoor dataset.

In Chapter 4, a novel deep convolutional neural network is presented. The network takes a pair of sequential images as input, and attempts to infer the absolute speed of the camera between them. The regressed speed is used to re-scale relative pose estimates computed from a monocular visual odometry method. The resultant odometry method operates scale drift-free over a large outdoor dataset.

In Chapter 5, a novel sparsified direct tracker is introduced. Patches centred around sparse corners are extracted from keyframes in the map. Camera tracking is achieved by warping these patches into the current camera frame and minimising the resultant photometric error. As pixels are dealt with in a localised region, an efficient affine warp is used for each corner in replace of a full projective one. Additionally, a naïve inverse additive formulation is proposed for a potential reduction in computational complexity. The reduced computational cost allows for multiple reference keyframes to be used for tracking, ensuring a globally consistent pose estimate. Estimated camera trajectories are equal if not more accurate than those obtained from a state of the art tracker that uses regions of high image-gradient.

In Chapter 6, a novel sparsified direct map optimisation is introduced. Keyframes and the depths of their associated corners are optimised in a bundle adjustment that, like the proposed tracker, aligns a sparse patches in the image. A full SLAM algorithm that utilises both the direct tracker and mapper is also introduced. A common alternative for sparsified direct methods is

depth-filtering. The method is compared to filtering in terms of structural estimation, where it often achieves greater accuracy, despite having more trouble with occlusion.

In Chapter 7, the results from the previous chapters are synthesised into the key contributions of this thesis. Building on the insights gleaned in this thesis, future work is finally suggested.

Chapter 2

Object-aware bundle adjustment

This chapter introduces a method for the correction of scale drift in monocular SLAM. Objects of a known class and size are incorporated into the map and used in a novel bundle adjustment to provide hints about absolute scale. The method is tested on a long range dataset and obtains results that are metrically accurate and scale drift-free.

2.1 Introduction

Being a form of dead-reckoning, and using error-prone sensor measurements in the estimation of a less than completely defined model of the world, any simultaneous localisation and mapping (SLAM) algorithm is destined to accumulate drift as the sensor moves further from its starting position.

Using range sensors, such as lidar, RGBD, and stereo cameras, this drift from reality has little impact on the ability of SLAM to continue working. With monocular visual SLAM however, the additional drift that occurs through the depth-scale/speed ambiguity can incapacitate the method entirely. For example, keyframe-based approaches often select keyframes based on their proximity to their nearest neighbour. If the estimate of speed is too large, keyframes will be created too rapidly, and if too small they are not created at all. Again, while all methods are able to redistribute and reduce accumulated error using loop-closure [39, 8, 40, 41, 42], scale drift in monocular methods can be so severe that searching for closure is infeasible. Some studies (*e.g.* [43]) also report difficulty rectifying scale drift some way into the process of map building, indicating that the chosen methods of accounting for and distributing error are irreversible.

Our own visual experience suggests that scale/speed drift does not occur for a cyclopean hu-

man observer moving through the environment [27]. With one eye shut, we can still draw on our wealth of prior scene knowledge to stabilise scale.

This chapter attempts to bring some of the same high level knowledge that humans possess to monocular SLAM. It is shown that a modified bundle adjustment can counter scale drift effectively by detecting and tracking object classes with a known size within the map. As a continuous estimate of scale is crucial to long term performance, drift is corrected frequently in local adjustments rather than in a more global correction. The additional computation is small compared with landmark-only bundle adjustment and relies on only modest prior information.

The principal approach offered in this chapter is developed in Section 2.5, where we detail a novel method for object-supplemented bundle adjustment, one which offers a combined representation for point landmarks and extended objects. Section 2.6 outlines its implementation and Section 2.7 gives results from experimentation using the KITTI dataset and sequences from handheld phone cameras. Finally, concluding remarks and directions for future research are presented in Section 2.8.

Before presenting the main contribution, Section 2.2 reviews methods for correcting scale drift and incorporating objects into SLAM, and Section 2.4 presents some preliminary work on incorporating constraints between pairs of landmarks. Although ultimately unused, it offers some insight into representational avenues (and cul-de-sacs).

2.2 Related work

Previous work in mitigating scale uncertainty in SLAM falls into one of two categories: either information from additional sensors is utilised or, like the method proposed here, some assumptions about the camera's environment are made.

A popular method of resolving scale is to employ additional sensors alongside the camera itself. A natural extension is of course a second camera [44] which immediately resolves scale if the baseline between the two cameras is known. A favoured non-visual sensor is the inertial measurement unit or IMU, which is able to measure its own acceleration and 3D orientation using a mix of accelerometers and gyroscopes. Jung and Taylor [45] introduce IMU measurements into a structure from motion method using spline fitting. Sets of keyframes are first divided into windows for a particular time frame. Each window has access to an estimated trajectory obtained

by double integrating the accelerometer measurements at the start of the window period. Rather than use keyframes to produce a trajectory estimate, splines are fitted to each window such that they agree both with keyframe positions and accelerometer readings. A common practice is to fuse both visual and IMU measurements in an EKF. Nützi *et al.* [46] fuse IMU information with the tracking output of Klein and Murray's PTAM [9], while Hide *et al.* [47] fuse it with up-to-scale relative pose estimates from a downward facing camera. Achtelik *et al.* [48] additionally incorporate pressure measurements into an the EKF, which is used to correct scale for a micro air vehicle (MAV).

A common assumption for long term visual odometry methods is that the camera is fixed at a known height above the ground plane. By detecting the ground plane, the local scale of the map may be corrected to ensure that this height stays at its known value [49, 50, 51, 52, 53]; these methods excel when this assumption holds, but are incapable of operating otherwise, e.g. with aerial images taken from a drone flying at variable altitudes. Detection of the plane can be particularly extensive, where objects likely to be resting on the ground are tracked throughout video to aid in the estimate [53].

A second assumption is that there is some repetitive structure to the environment being mapped. One such case is the method proposed by Botteril *et al.* [54] where pairs of neighbouring landmarks are considered objects by the method. An object's size, which corresponds to the distance between its constituent landmarks, is learnt online. When a previously seen object is seen again, its expected size may be used to measure and correct scale drift. As no prior knowledge is used, the method does not permit absolute scale to be recovered, rather for more certain parts of the map to impart this information on others without actual loop closure.

Strasdat *et al.* describe a pose-graph optimisation that claims to be scale drift-aware [55]. While pose-graph optimisation has a much lower computational complexity than full bundle adjustment, it is unable to correct scale drift on loop closure. The method proposed is an effort to allow scale correction in pose-graph optimisation by placing similarity rather than rigid-body constraints between pairs of keyframes. When a loop is detected, drift is measured by comparing overlapping structure. This information is fed into the scale parameter of the loop-closure constraint, which is propagated to the rest of the loop after optimisation. While the method does show some improvement over rigid body pose-graph optimisation, the question of whether the method is better than a full bundle adjustment is left unanswered. More importantly, scale is only corrected

at loop-closure, rendering it useless for open-loop camera trajectories.

Castle *et al.* [43] incorporate planar objects identified by a set of unique SIFT features [56] into a map estimated using monoSLAM [4, 5]. Objects are detected live, and a homography between the plane and the camera is estimated and decomposed, allowing the object to be incorporated into monoSLAM's EKF. As the scale of each object instance is known, the method is able to resolve depth/speed scaling ambiguity, but the requirement for specific objects makes broader application unlikely. Despite being able to set a global scale from object measurements, correcting scale-*drift* using object measurements proved problematic in an EKF. Object measurements incorporated at later parts of the map are increasingly at odds with the scale of the surrounding landmarks. This resulted either in tracking failure or rejection of object detections. In later work [57] they use a bundle adjustment rather than an EKF, but objects were merely used as augmentations rather than for scale correction. Civera *et al.* [58] also used an EKF and augmented their map with full three-dimensional objects; but they too considered particular objects rather than object classes.

Two further works that make use of objects to improve map accuracy are [59] and [60]. Bao *et al.* [59] jointly estimate camera, point and rectangular object positions in a bundle adjustment. However, the focus of their work is on increasing accuracy in structure from motion rather than correcting scale over long term SLAM trajectories. Gálvez-López *et al.* [60] recently proposed a method that adds objects modelled from point clouds to the map in which known distances between object-points are used for adding additional geometrical constraints to a bundle adjustment and enforcing scale in the map. As the method uses a library of specific object instances, its applicability in a more general setting is uncertain. Although the global scale of the map is metrically accurate, no results for long-term drift-free operation are shown.

Dame *et al.* [61] use a monocular method to produce a dense map that is later refined using 3D shape priors embedded in a low-dimensional latent space. As the scale of shapes is known, the scale of the map may be set. Like [60], however, the method has only been tested in a small localised map rather than long range data and it is unclear if objects would be able to correct drift rather than setting a global scale.

The method presented in this chapter draws on methods that supplement the map with objects, in preference to using some form of assumption or additional hardware. Rather than incorporate specific object instances, however, the method takes advantage of an entire object class. Object classes are chosen based on their ubiquity and the variance of their actual size. Naturally, a low

variance is desirable. A minimal object representation is used for speed, and objects are actually incorporated into the map and used directly rather than being used to localise a ground plane. Rather, scale is estimated online in a local bundle adjustment optimisation with little prior assumptions on the environment other than it contains the classes in question. Object size distributions are learnt beforehand to avoid drift in the estimation and thereby guarantee long-term accuracy and ensure a metrically accurate scale.

2.3 Sparse Monocular SLAM Representation

Sparse SLAM maps are represented by point-landmarks in three dimensional space. A landmark $\mathbf{X}_{jw}$ is parameterised by its homogeneous 3D coordinates in the world w

$$\mathbf{X}_{jw} = [X_{jw}, Y_{jw}, Z_{jw}, 1]^\top \ . \tag{2.1}$$

The camera's pose within the world is represented by a Euclidean transformation $\mathtt{T}_i \in \mathbf{SE}(3)$

$$\mathtt{T}_i = \begin{bmatrix} \mathtt{R}_i & \mathbf{t}_i \\ \mathbf{0} & 1 \end{bmatrix} \ , \tag{2.2}$$

which is composed of the rotation matrix $\mathtt{R}_i \in \mathbf{SO}(3)$, where $\mathbf{SO}(3)$ is the group of rotations in 3D euclidian space, and a translation vector $\mathbf{t}_i \in \mathbb{R}^3$. The matrix $\mathtt{T}_i$ transforms landmark's coordinates from world-centric coordinates $\mathbf{X}_{jw}$ to camera-centric coordinates $\mathbf{X}_{ij}$ where

$$\begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} = \mathtt{T}_i \begin{bmatrix} \mathbf{X}_{jw} \\ 1 \end{bmatrix} \ . \tag{2.3}$$

The estimated projection $\hat{\mathbf{x}}_{ji}$ of the landmark in the camera frame is computed as

$$\begin{bmatrix} \hat{\mathbf{x}}_{ji} \\ 1 \end{bmatrix} = \mathtt{K}\mathrm{proj}(\mathbf{X}_{ijw}) \ , \tag{2.4}$$

where $\mathrm{proj}(\,)$ is the projection function

$$\mathrm{proj}(\mathbf{X}_{ij}) = \begin{bmatrix} X_{ij}/Zij \\ Y_{ij}/Zij \\ 1 \end{bmatrix}, \tag{2.5}$$

where $\mathtt{K}$ is the camera's intrinsic calibration matrix

$$\mathtt{K} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

with focal lengths f_u and f_v and principal point $[u_0, v_0]$. We define the mapping from $\mathbf{X}_{jw}$ to $\hat{\mathbf{x}}_{ji}$ as the combined projection function $\mathrm{proj}_K(\,)$

$$\hat{\mathbf{x}}_{ji} = \mathrm{proj}_K(\mathbf{X}_{jw}, \mathtt{T}_i) = \mathrm{proj}_K(\mathbf{X}_{ij}) \tag{2.7}$$

Similarly we define the combined back-projection function $\mathrm{proj}_K^{-1}(\,)$ which maps from $\hat{\mathbf{x}}_{ji}$ to $\mathbf{X}_{jw}$. As proj_K is non-invertable, this requires an additional argument, the depth of $\hat{\mathbf{x}}_{ji}$ in $\mathtt{T}_i$, d_j.

$$\mathbf{X}_{jw} = \mathrm{proj}_K^{-1}(\hat{\mathbf{x}}_{ji}, d_j, \mathtt{T}_i) \tag{2.8}$$

2.4 Landmark-to-landmark constraints

Accepting that object size can regulate scale does not answer the question of how best to incorporate size information into a bundle adjustment. A first proposal tested here is that objects provide a metrically accurate coordinate system on which expected distance *between* landmarks may be computed. These landmark-to-landmark (L2L) constraints will be used to regularise bundle adjustment.

Consider a set of landmarks $\mathcal{X} = \{\mathbf{X}_{0w}, \mathbf{X}_{1w}..., \mathbf{X}_{Jw}\}$ which are observed in a set of video frames with poses $\mathcal{T} = \{\mathtt{T}_0, \mathtt{T}_1, ..., \mathtt{T}_I\}$.

Bundle adjustment seeks to find landmark positions and camera poses that minimise the reprojection error, the summed weighted 2-norms of the errors $\mathbf{r}_{ji} = (\mathbf{x}_{ji} - \hat{\mathbf{x}}_{ji}(\mathbf{X}_{jw}, \mathtt{T}_i))$ between

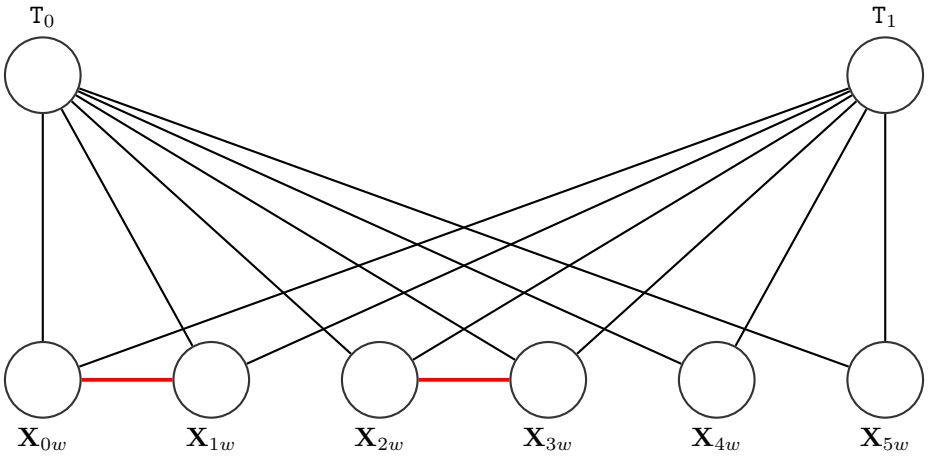


Figure 2.1: MRF of a bundle adjustment with L2L constraints. Scale dependent terms are shown in red.

between image measurements $\mathbf{x}_{ji}$ and their predicted projections $\hat{\mathbf{x}}_{ji}$. Assuming errors are normally distributed with zero mean and covariance $\mathtt{W}_{ji}$, bundle adjustment provides the maximum likelihood estimator of $\mathcal{X}$ and $\mathcal{T}$:

$$\{\mathcal{X}, \mathcal{T}\} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min}\, E_{\text{reproj}} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min} \sum_{i\in\mathcal{T}} \sum_{j\in\mathcal{X}} \mathbf{r}_{ji}^{\top} \mathtt{W}_{ji}^{-1} \mathbf{r}_{ji} \,. \tag{2.9}$$

Bundle adjustment with L2L constraints aims to find the landmarks and keyframes that minimise reprojection error regularised by an additional error that measures the degree to which the constraints are satisfied

$$\{\mathcal{X}, \mathcal{T}\} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min} \left[E_{\text{reproj}} + \lambda E_{\text{constraint}}\right]. \tag{2.10}$$

The constraint term supposes that the distance between two landmarks j and j' is drawn from some known normally distributed prior with mean $\mu_{jj'}$ and variance $\sigma^2_{jj'}$. Writing the estimated distance between the pair of landmarks $\mathbf{X}_{jw}$ and $\mathbf{X}_{j'w}$ as $d_{jj'}$,

$$E_{\text{constraint}} = \sum_{j,j'\in\mathcal{X}} \sigma^2_{jj'} (d_{jj'} - \mu_{jj'})^2 \,. \tag{2.11}$$

The MRF representation of a bundle adjustment with constraints between landmarks is shown in Fig. 2.1, where red edges represent the constraints between landmarks.

2.4.1 Experimental results

The approach was first tested in simulation. As illustrated in Fig. 2.2 a virtual camera was moved along a circular path in an anti-clockwise direction around a scene of 200 landmarks drawn from a cylindrical distribution and shown by blue crosses in the figure. Keyframes were added at equally-spaced distances. Each time a new keyframe was added, a local bundle adjustment was performed. This adjustment included the new keyframe, its three nearest neighbouring keyframes and the landmarks visible in all of them. The path and keyframes (shown as circles) are highlighted in red. Perfect data association was assumed, but the measurements were corrupted with zero-mean Gaussian noise with a standard deviation of 1 pixel.

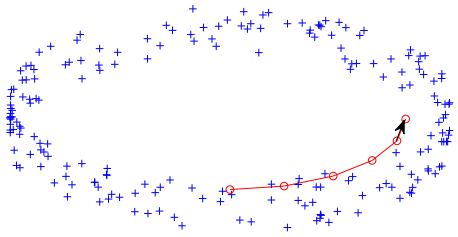
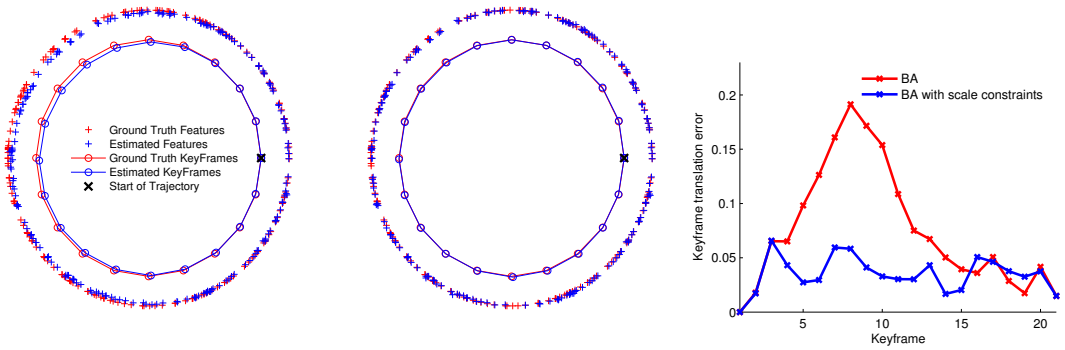


Figure 2.2: Simulation environment. The blue crosses are virtual landmarks drawn from a cylindrical distribution. The camera moves in an anti-clockwise direction within shown by the black arrow. Keyframes are shown as red circles.

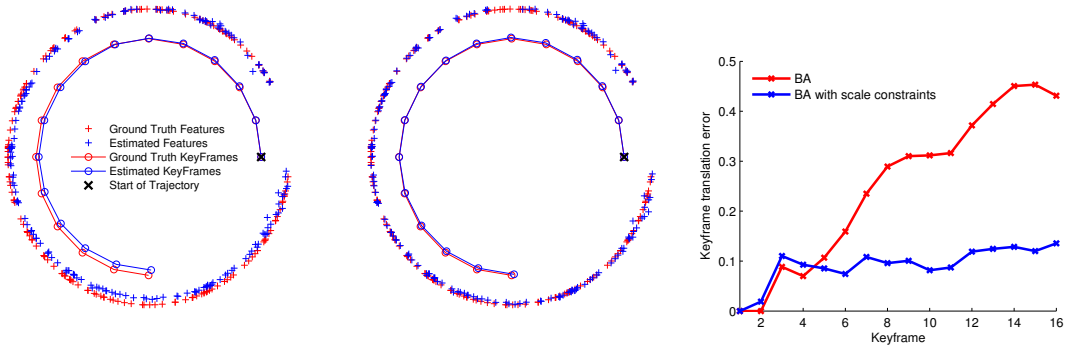
At the end of a trajectory both constrained L2L and unconstrained bundle adjustments were performed. In the former, a single L2L constraint was added per keyframe by picking two random landmarks visible in each and using the ground-truth distance between them as the expected distance. In the latter, to fix the scale gauge for ease of comparison, the distance between the first and second keyframes is reset to its ground-truth value each iteration. (This need not be done in the constrained bundle adjustment as scale is implicitly enforced by constraints.)

Closed-loop results. Figs. 2.3(a,b) show views of the reconstructed camera trajectory (blue) and landmarks compared to their ground-truth counterparts (red) using (a) conventional bundle adjustment and (b) L2L bundle adjustment, *after loop closure*. In this situation the difference between two results is slight, though the L2L adjustment is clearly superior. Fig. 2.3(c) shows the error



(a) Unconstrained (b) L2L constrained (c)

Figure 2.3: Results from (a) unconstrained and (b) constrained adjustments for a closed-loop trajectory. (c) Per-keyframe translation errors compared.



(a) Unconstrained (b) L2L constrained (c) Translation errors

Figure 2.4: Results from (a) unconstrained and (b) constrained adjustments for an open-loop trajectory. (c) Per-keyframe translation errors compared.

in translation per keyframe when compared to the ground-truth data. (This error is calculated as the Euclidean norm of the distance between the reconstructed and ground-truth keyframes' translation and Euler-angle rotation parameters respectively. For a ground-truth keyframe pose $\mathtt{T}_i$ the translation error of its corresponding estimation $\hat{\mathtt{T}}_i$ is defined as

$$E_{\text{trans}}(\hat{\mathtt{T}}_i) = \sqrt{(\mathbf{t}_i - \hat{\mathbf{t}}_i)^\top (\mathbf{t}_i - \hat{\mathbf{t}}_i)}\,, \tag{2.12}$$

where $\mathbf{t}_i$ is the translational component of $\mathtt{T}_i$. The results confirm that L2L constraints are able to reduce per-keyframe error across the trajectory. The reduction in translation error is particularly noticeable, and is greatest at keyframes furthest from loop closure for bundle adjustment. The error is kept at a constant level throughout when using L2L constraints.

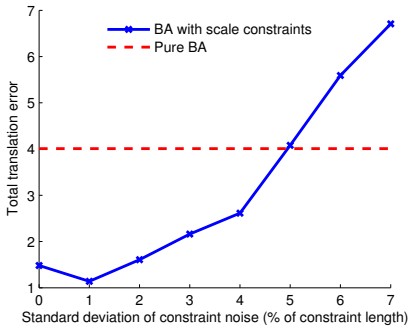
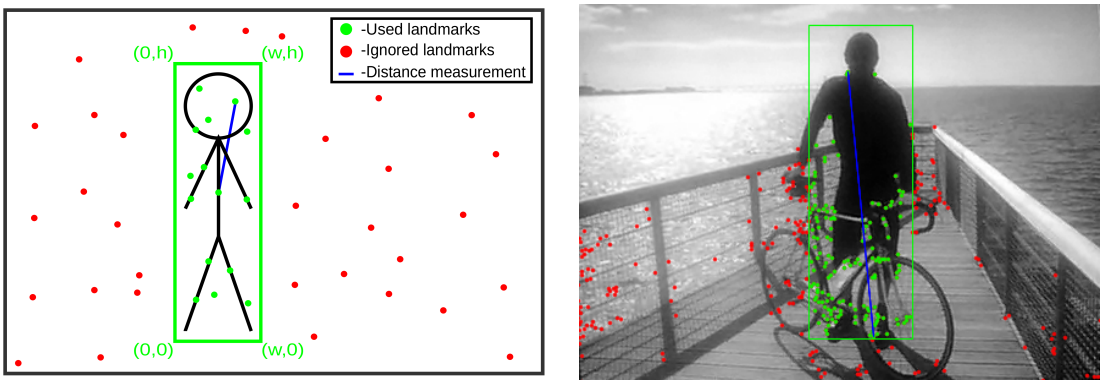


Figure 2.5: Total keyframe translational error for varying standard deviation of constraint noise.

Open-loop results. The method was tested on the same trajectory, but now without loop closure. Fig. 2.4 shows the comparison. The use of L2L constraints now gives more clear benefit. Fig. 2.4(c) shows again that while translational error accumulates as the camera moves further from the start of the trajectory, the method using L2L constraints keeps this error close to constant throughout.

Adding noise to the constraints. The results in Figs. 2.3 and 2.4 rely on L2L constraints which have perfect knowledge of the distance between individual landmarks. To introduce uncertainty, each L2L constraint was drawn from a Gaussian distribution centred around its ground-truth value. The standard deviation was chosen as a percentage of the constraint length so as not to penalise constraints with smaller lengths. Fig. 2.5 shows the total error in keyframe translation for various percentages. L2L-constrained adjustment can tolerate noise with a standard deviation of up to 5% of their length before being outperformed by bundle adjustment without constraints.

L2L constraints from object detection in imagery. Obtaining L2L constraints from imagery involves detecting known object classes in keyframes as they are added. A part-based object detector [62] using histograms of oriented gradients [63] was used, although any detector is acceptable. In each keyframe containing detected objects L2L constraints were generated using the process illustrated in Fig. 2.6. A bounding box coordinate system is thrown round the detection and, from pre-computed distributions on the object's width and height, and assuming planarity, distance constraints between landmarks within the object obtained by projection. Assuming height is normally distributed as $h \sim \mathcal{N}(\mu_h, \sigma_h^2)$, and width is some proportion of height $w = \beta h$, the distance between two landmarks with box-coordinates (u_1, v_1) and (u_2, v_2) respectively is calculated



(a) (b)

Figure 2.6: Object detection for constraint calculation. (a) A person detected in the image. The landmarks in red are ignored while the landmarks in green that lie within the bounding box are used. The bounding box sets up its own coordinate frame within the image in which expected distances between points may be calculated. An example distance between a pair of points is highlighted in green. (b) Example from imagery.

using

$$d(u_1, v_1, u_2, v_2) = \sqrt{h^2(u_1 - u_2)^2 + w^2(v_1 - v_2)^2} \tag{2.13}$$

$$= h\sqrt{(u_1 - u_2)^2 + \beta^2(v_1 - v_2)^2)}\,. \tag{2.14}$$

The constraint in Eq. (2.11) is then normally distributed with mean

$$\mu_{jj'} = \mu_h\sqrt{(u_1 - u_2)^2 + \beta^2(v_1 - v_2)^2)} \tag{2.15}$$

and variance

$$\sigma^2_{jj'} = \sigma^2_h\left[(u_1 - u_2)^2 + \beta^2(v_1 - v_2)^2\right]\,. \tag{2.16}$$

An example of constraint calculation is shown in Fig. 2.6(b), where the prior distance between two landmarks is calculated to be $1.56\,\mathrm{m}$, an entirely reasonable value for the distance between shoulder and foot in a male.

In a live system. The L2L regulariser was tested by modifying the basic Oxford PTAM system. It became immediately clear that the use of constraints in this way was problematic. First, the planar assumption needed to obtain the constraints proved highly restrictive. Second, constraint information was introduced some time into the processing and once the scale had drifted, was

unable to correct the drift. Landmarks with constraints would indeed adhere to them, but the consequential large changes in their position often resulted in the bundle adjustment ignoring their measurements as outliers rather than re-scaling the map.

2.4.2 Discussion

The difficulties experienced with L2L regularisers was not unexpected. Our findings are consistent with Castle *et al.* [43] who incorporated planar objects into an EKF and attempted to impose inter-landmark constraints from them. This suggests that sudden interventions at low-level are too disruptive of the probabilistic basis guiding the optimisation, whether recursively via a Kalman filter or in batch mode via bundle adjustment; and both suggest avoiding transference of somewhat coarse-scale information object size directly into fine-scale point landmarks already in the map. Instead we propose representing both objects and sparse 3D points as generalised landmarks with different scales or "extents".

2.5 Object bundle adjustment using landmarks with extent

Rather than using regularisation, where balancing the cost of minimising reprojection error with that of obeying pairwise constraints proves awkward, we now develop an approach where both point landmarks and finite-sized objects share the same representation. The representation of scene and image is discussed in Section 2.5.1.

Section 2.5.2 considers the process of detecting objects and solving data association between detections, and the core aspects of how detections are used to instantiate objects in the map and therefore incorporated into the bundle adjustment are given in 2.5.3.

Section 2.5.4 details how the new bundle adjustment is used to locally correct scale as the camera explores new parts of the environment. Outliers, false detections and other model violations are considered in Section 2.5.5.

2.5.1 The representation of scene and observation

A number of object parameterisations have been introduced in SLAM before, ranging from bounding boxes [64, 65] to more sophisticated full 3D surface models [61]. While more complicated models certainly allow for accurate object segmentations, they impose a larger computational over-

head. Even simple rectangular bounding boxes require a full 6DOF pose to be maintained, while 3D surface models require 2D silhouette segmentation to be localised correctly. While the results of such localisations certainly are visually impressive, we argue that their additional complexity is *unnecessary* if their sole purpose is scale correction. It is intuitive that constraining a single gauge only requires a single scale-dependent measurement. For this reason, we propose using an object parameterisation that is almost identical to that of landmarks.

Objects are represented by a minimum enclosing sphere, and have a single additional scale-dependent parameter, the sphere's radius. We shall call this the *extent*. An object is thus represented in the world frame by

$$\mathbf{Q}_{kw} = [\mathbf{X}^\top, \epsilon\,]^\top_{kw}\,, \tag{2.17}$$

where $\mathbf{X} = [X, Y, Z, 1]^\top$ is the homogeneous location of its centre and ϵ its extent. The object parameterisation is shown in Fig. 2.7 alongside that for a point landmark.

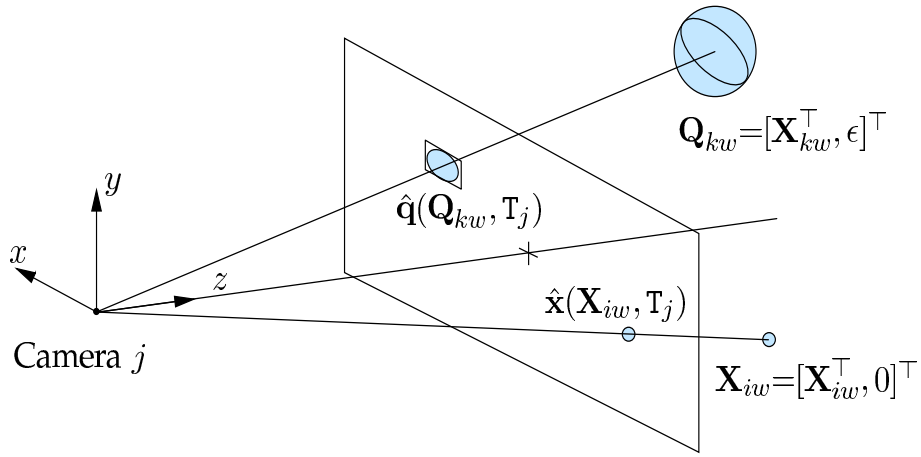


Figure 2.7: Parameterisations of landmarks and objects and their respective projections.

As the object is deemed spherical, transforming between coordinate frames is simple. An object k in world-coordinate space $\mathbf{Q}_{kw}$ has the coordinates $\mathbf{Q}_{ki}$ in frame-coordinate space given by

$$\mathbf{Q}_{ki} = \begin{bmatrix} \mathbf{X} \\ \epsilon \end{bmatrix}_{ki} = \begin{bmatrix} \mathtt{T}_i & \mathbf{0}^{4\times1} \\ \mathbf{0}^{1\times4} & 1 \end{bmatrix} \mathbf{Q}_{kw}\,, \tag{2.18}$$

where $\mathtt{T}_i$ is the pose of the ith frame. The projection of the object in the frame i is the 4-vector

$$\hat{\mathbf{q}}_{ki}(\mathbf{Q}_{kw}, \mathtt{T}_i) = [u, v, w, h]^\top_{ki}\,, \tag{2.19}$$

where $[u, v]$ are the image coordinates of the projection of the centre of the object so that

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathtt{K}\ \mathrm{proj}(\mathbf{X}) \ , \tag{2.20}$$

and where $\mathtt{K}$ is the camera's intrinsic matrix, with focal lengths f_u and f_v and principal point $[u_0, v_0]$. The parameters $[w, h]$ are the width and height of the projected bounding box in the image and are given by

$$\begin{bmatrix} w \\ h \end{bmatrix}_{ki} = \frac{2\epsilon_{ki}}{Z_{ki}} \begin{bmatrix} f_u \\ f_v \end{bmatrix} \ . \tag{2.21}$$

2.5.2 Object measurements and data association

Objects are localised using a detector applied to each new keyframe as it is created. Object measurements are determined from the bounding box from the detector, and are parameterised in the same way as object reprojections by the 4-vector $\mathbf{q}_{ki} = [u, v, w, h]^\top_{ki}$. The projections of objects of known extent are immediately invertible, and an initial location could be entered into the map from just a single measurement. However, the bundle adjustment developed below treats objects in the same way as landmarks, and object detections are required in a number of keyframes, and so data association must be solved. Here we use [66, 67] but the choice is not critical. Note that we prefer to keep object tracking independent of the map, preventing bad estimates of the latter from negatively influencing the former.

Example detections from two adjacent keyframes are shown in Fig. 2.8.

2.5.3 Object bundle adjustment

The incorporation of a unified representation of landmarks and objects into bundle adjustment is now developed.

Recall that bundle adjustment based on point landmarks alone finds the sets of landmarks $\mathcal{X}$ and keyframe poses $\mathcal{T}$ that minimise the reprojection error

$$\{\mathcal{X}, \mathcal{T}\} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min}\, E_{\mathrm{reproj}} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min} \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{X}} \mathbf{r}^\top_{ji} \mathtt{W}^{-1}_{ji} \mathbf{r}_{ji} \ ,$$

Figure 2.8: Object detections from two adjacent keyframes. Detections and their associated labels act as measurements for objects.

where $\mathbf{r}_{ji}$ is the difference in measured and estimated landmark re-projection, and $\mathtt{W}_{ji}$ is the estimated measurement noise covariance.

When an object k is detected in a keyframe with pose $\mathtt{T}_i$ it will produce an image measurement $\mathbf{q}_{ik}$. If object measurements are similarly distributed with covariance $\mathtt{V}_{ki}$ a new bundle adjustment that seeks to find the most likely set of keyframes, point landmarks, and objects $\mathcal{Q} = \{\mathbf{Q}_{0w}, \mathbf{Q}_{1w} ..., \mathbf{Q}_{Kw}\}$ as $\{\mathcal{T}, \mathcal{X}, \mathcal{Q}\}$ may be written as

$$\{\mathcal{T}, \mathcal{X}, \mathcal{Q}\} = \underset{\{\mathcal{T},\mathcal{X},\mathcal{Q}\}}{\arg\min} \left(\sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{X}} \mathbf{r}_{ji}^{\top} \mathtt{W}_{ji}^{-1} \mathbf{r}_{ji} + \sum_{i \in \mathcal{T}} \sum_{k \in \mathcal{Q}} \mathbf{r}_{ki}^{\top} \mathtt{V}_{ki}^{-1} \mathbf{r}_{ki} \right), \tag{2.22}$$

where $\mathbf{r}_{ki} = (\mathbf{q}_{ik} - \hat{\mathbf{q}}_{ki}(\mathbf{Q}_{kw}, \mathtt{T}_i))$.

The MRF representation of the object-supplemented bundle adjustment is shown in Fig. 2.9, with scale dependent variables highlighted in red.

However, the object-supplemented bundle adjustment can be written more economically. We first assume a known extent for a particular object class, and fix the extent of each object instance of the class to this value: ϵ becomes a hyper-parameter of the method. As the extent is fixed prior to operation, only the object's position in the map are refined.

Second, we treat landmarks as objects of zero extent, making $\mathcal{X}$ a subset of $\mathcal{Q}$. Eq. (2.22)

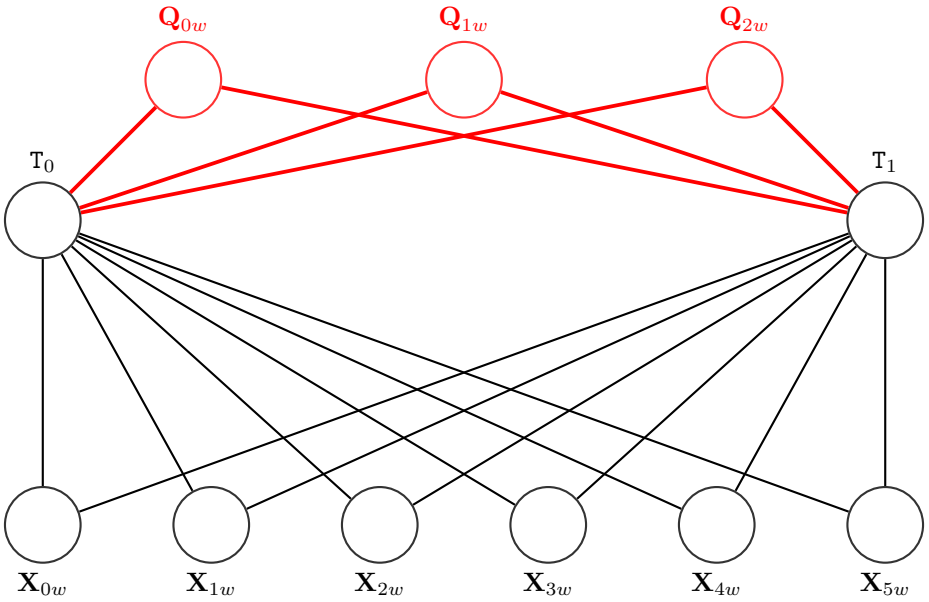


Figure 2.9: MRF representation of object-supplemented bundle adjustment. Scale-dependent variables and constraints are shown in red.

becomes

$$\{\mathcal{Q}, \mathcal{T}\} = \underset{\{\mathcal{Q},\mathcal{T}\}}{\arg\min} \sum_{i \in \mathcal{T}} \sum_{k \in \mathcal{Q}} \mathbf{r}_{ki}^{\top} \mathtt{V}_{ki}^{-1} \mathbf{r}_{ki} \ . \tag{2.23}$$

This simplification permits the immediate use of any sparse Levenberg-Marquardt algorithm to be used for optimisation. One such an example may be found in Appendix 6 of [68]. Although there are additional operations to deal with the extra parameters in the measurement residuals, the computational complexity is identical to bundle adjustment using landmarks alone.

The covariance matrices for landmarks and objects remain distinct to deal with the larger uncertainty in object measurements. With a known noise model for the object detection algorithm, it is possible to calculate $\mathtt{V}_{ki}$ for objects. If an object's extent is normally distributed as $\mathcal{N}(\epsilon, \sigma_\epsilon^2)$, and using Eq. 2.21, its reprojection in an image of frame k will also be normally distributed as

$$\begin{bmatrix} w \\ h \end{bmatrix} \sim \mathcal{N}\left(\epsilon\, \mathbf{f}, \sigma_\epsilon^2\, \mathbf{f}\, \mathbf{f}^{\top}\right) \ , \tag{2.24}$$

where

$$\mathbf{f} = \frac{2}{Z_{ki}} \begin{bmatrix} f_u \\ f_v \end{bmatrix} \ . \tag{2.25}$$

Assuming zero-mean Gaussian noise on the width and height and a covariance of Σ_{bb}, the measured

width and height are also normally distributed as

$$\begin{bmatrix} \hat{w} \\ \hat{h} \end{bmatrix} \sim \mathcal{N}\left(\epsilon\,\mathbf{f}, \sigma_\epsilon^2\,\mathbf{f}\,\mathbf{f}^\top + \Sigma_{bb}\right) , \tag{2.26}$$

and assuming independent zero-noise Gaussian noise on the centre of the bounding box, the final covariance for object measurements is

$$\mathtt{V}_{ki} = \begin{bmatrix} \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix} & \mathbf{0}^{2\times 2} \\ \mathbf{0}^{2\times 2} & \sigma_\epsilon^2\,\mathbf{f}\,\mathbf{f}^\top + \Sigma_{bb} \end{bmatrix} , \tag{2.27}$$

where $\sigma_{x,y}^2$ are variances on the detection centre. For landmarks, the bottom right 2×2 matrix collapses to zero.

2.5.4 Tracking and local bundle adjustment

Applying a global bundle adjustment becomes computationally infeasible as the map grows. Instead, we optimise a local area around the current estimated camera position, giving a broadly constant computational load consistent with video rate operation. For the current camera pose $\mathtt{T}_{\text{cam}}$, a local set of n (10 in our case) nearest keyframes $\mathcal{T}_{\text{local}}$ is defined. The structure to be refined consists of the local set of landmarks and objects visible in these keyframes $\mathcal{Q}_{\text{local}}$. To constrain the local adjustment to the rest of the map the set of keyframes $\mathcal{T}_{\text{fixed}}$ which have observations of $\mathcal{Q}_{\text{local}}$ is also included as fixed keyframes in the optimisation. The optimisation is

$$\{\mathcal{Q}_{\text{local}}, \mathcal{T}_{\text{local}}, \mathtt{T}_{\text{cam}}\} = \underset{\{\mathcal{Q}_{\text{local}}, \mathcal{T}_{\text{local}}, \mathtt{T}_{\text{cam}}\}}{\arg\min} \sum_i \sum_k \mathbf{r}_{ki}^\top \mathtt{V}_{ki}^{-1} \mathbf{r}_{ki} , \tag{2.28}$$

where $i \in \{\mathtt{T}_{\text{cam}}, \mathcal{T}_{\text{fixed}}, \mathcal{T}_{\text{local}}, \}$ and $k \in \mathcal{Q}_{\text{local}}$. Because the adjustment is run in only a local area around the current camera position, it is unable to propagate corrections from the local window to the rest of the map. If scale drift occurs due to a lack of object detections over a period, only the most recent portion of the map consisting of the n keyframes in the local window will be corrected when objects are seen again.

2.5.5 Outlier rejection

Object measurements are subject to a number of sources of error, namely: (i) detections have inaccurate size and location in the image which affects the accuracy of the position of the 3D world model estimated by the local bundle adjustment; (ii) false positives in object detection; (iii) incorrect association between otherwise correct detections; and lastly and most importantly (iv) moving objects can lead to grossly inaccurate scale estimations if they are allowed to be used in the map. These four may be reduced using two strategies described here.

Minimum number of object detections. Unlike landmarks — which are essential for camera localisation and must be localised and used as soon as possible — objects need not be used so promptly. Rather, we impose a minimum number of measurements on each object before it is allowed to be used in optimisation. The benefits of this are threefold. First, the increased number of measurements ensures only well-localised objects are used. Secondly, any false positives from the object detection algorithm should be ignored provided they fall under this threshold. Finally, unless they are in front of the camera and moving at the same speed, moving objects will have less than the required number of measurements and will consequently be ignored. For testing we use a minimum of 20 measurements. A lower threshold allows more objects to be included which may aid with scale information at the potential expense of map corruption should an object be moving.

Robust estimator. To down-weight poor object measurements or bad data associations, object and point landmark measurements are wrapped in a Tukey biweight objective function (c.f. [69]). Eq. (2.23) becomes

$$\{\mathcal{Q}, \mathcal{T}\} = \underset{\{\mathcal{Q},\mathcal{T}\}}{\arg\min} \sum_{i \in \mathcal{T}} \sum_{j \in \mathcal{Q}} \left\| \mathbf{r}_{ki}^{\top} \mathtt{V}_{ki}^{-1} \mathbf{r}_{ki} \right\|_{\sigma_T} , \tag{2.29}$$

where $\|\cdot\|_{\sigma_T}$ is

$$\left\| r^2 \right\|_{\sigma_T} = \begin{cases} \frac{\sigma_T^2}{6}(1 - [1 - \frac{r^2}{\sigma_T^2}]^3) & \text{for } r^2 \leq \sigma_T^2, \\ \frac{\sigma_T^2}{6} & \text{otherwise.} \end{cases} . \tag{2.30}$$

We consider three regimes. (i) For point landmark measurements, σ_T is set to the estimated standard deviation of the distribution of point errors. (ii) Object measurements are generally less accurate than landmark measurements, and σ_T is set to 10 pixels for the *centre* component of object reprojection error. (iii) If objects are not seen for a long period of time, it is possible that scale will

drift. In this case, estimated objects will be significantly larger or smaller than the surrounding map, resulting in high error in the boundary component of object residuals $\mathbf{r}_{ki}$. To allow the bundle adjustment to correct scale in this case, rather than rejecting measurements as outliers, no robust estimator is used for boundary components. As the system assumes a fixed world, an object moving at the same speed and direction as the camera incorrectly implies a stationary camera which could lead to a corrupt map. With sufficient additional measurements, however, moving objects will be considered outliers.

2.6 Implementation outline

The proposed combined representation of point and object landmarks can be applied to any keyframe-based SLAM system. Here we use Klein and Murray's PTAM [9], albeit with substantial modifications to permit operation on outdoor sequences. PTAM uses a velocity model to predict the pose of the current frame given its predecessor. It was found that a more viable alternative was to model speed rather than velocity. The actual direction of the camera from frame to frame was determined using a set of sparse feature matches between the two. Additionally, the search window used for PTAM's coarse pose estimate was increased to deal with the large image changes between each frame.

After initialisation, the tracker runs continuously every frame, using a simple motion prediction to assist matching of FAST features [70] to landmark projections and thence to iteratively optimise the pose using a re-weighted least squares algorithm. Like PTAM, the method matches features using 8×8 pixel templates that are first coarsely matched with patches from other features in a search radius and then iteratively refined for sub-pixel precision.

To determine whether a new keyframe is required, in addition to measuring the distance from other keyframes the entropy ratio between the current camera estimate and the most recent keyframe is monitored, as suggested by Kerl *et al.* [71]. The ratio α is computed as

$$\alpha = \frac{H(\mathtt{T}_{k:k+j})}{H(\mathtt{T}_{k:k+1})}\,, \tag{2.31}$$

where $H(\mathtt{T}_{k:k+j})$ is the entropy for the motion estimate from the last keyframe k to the current frame j, $H(\mathtt{T}_{k:k+1})$ is the entropy from keyframe k to the very next frame $k+1$. The entropy

for a motion estimate is considered proportional to the natural logarithm of the determinant of the covariance matrix Σ for the tracking error i.e.

$$H(\mathtt{T}_{j:k}) \propto \ln(|\Sigma_{j:k}|) \; . \tag{2.32}$$

This, unlike the distance metric, has the added benefit of being invariant to scale drift. If either metric crosses a threshold, the current frame is used as a keyframe.

As only a subset of landmarks is used for tracking, the mapmaker first searches for measurements of any other landmarks and adds them to the keyframe. A set of epipolar matches is found with the closest neighbouring keyframe using PTAM's patch-matching algorithm, which is used to triangulate new map points. Any object measurements (in the form of bounding boxes from object detections) are then added to the keyframe. If an object has not been seen before, it can be localised immediately from a single measurement in the keyframe, however it is not allowed into the bundle adjustment until its measurements exceed the threshold discussed in Section 2.5.5.

If there are enough object measurements in the surrounding keyframes, a local landmark and object bundle adjustment is applied. If no objects are found, or the currently visible ones do not have enough measurements, a local landmark-only bundle adjustment is performed. With a window of 10 adjustable keyframes the bundle adjustment takes around 150ms to converge and so tracking is not disrupted. The new keyframe is then added to a queue in the mapmaker, which adds it to the map in a separate thread. Finally the motion model is updated with the current position of the camera.

Unlike PTAM's mapmaker, the method does not perform any global or local optimisation, but is simply responsible for adding keyframes. This is mainly due to the low frame-rate of the dataset which will be used. The low frame-rate results in very frequent addition of keyframes to the map, giving a parallel mapmaker little chance to actually run a bundle adjustment.

2.7 Experiments and results

The performance of the method has been evaluated on **kilometre long** outdoor sequences from the KITTI street scene dataset [72, 50], and on 100-metre long outdoor sequences from hand held phone cameras.

Summary experiment. In summary, though the method is able to deal with multiple arbitrary object classes, the nature of the KITTI dataset make cars the obvious choice for a object class. The extent for cars was set at $\epsilon = 1.2\,\text{m}$, an average from the model ranges of several popular manufacturers. (This value is validated later in Section 2.7.5.) Object detections in keyframes and corresponding data association labels are obtained using the tracking-by-detection algorithm of Zhang *et al.* and Geiger *et al.* [73, 74]. Their method is tailored to the detection of vehicles in street settings and hence well suited for experimental evaluation using KITTI. It is assumed that point landmark measurements are independent and corrupted with zero Gaussian additive noise with a standard deviation of 1 pixel. Object measurement noise was assumed to be independent and have a standard deviation of 1 pixel for all components. (This is revisited in Section 2.7.6 where methods for obtaining a full distribution for an object class's extent are considered.) An overall examplar of the performance on on one KITTI sequence is shown in Fig. 2.10. Part (a) show a keyframe with both point and objects and (b) shows the overall map.

Detailed experiments. In detail, individual experiments are reported as follows.

1. Section 2.7.1 compares performance of object bundle adjustment with an otherwise identical one that ignores objects and thus has no information about scale.

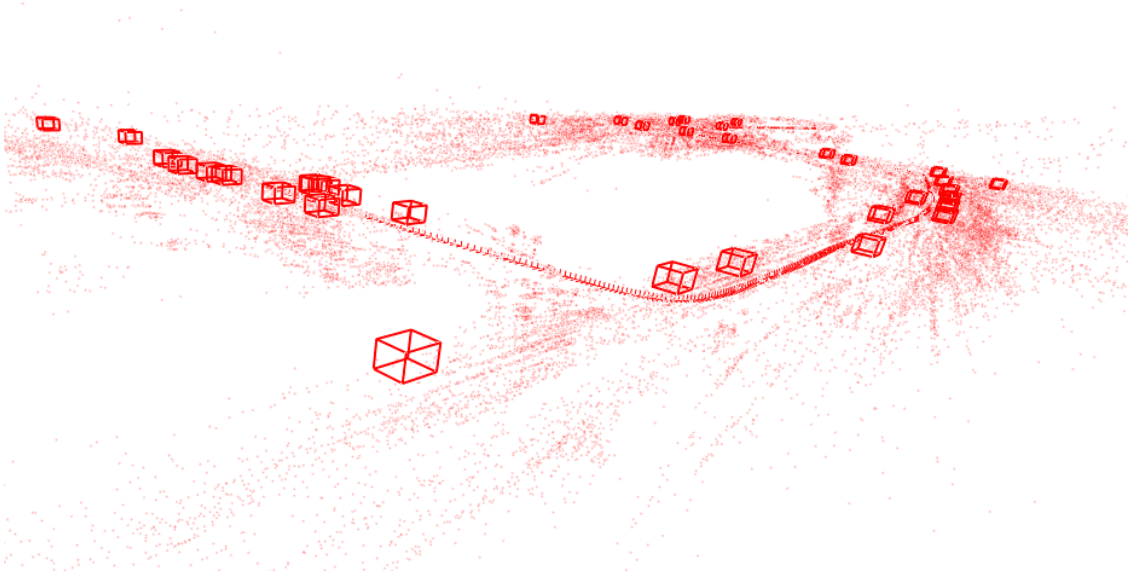
2. In Section 2.7.2, the performance of the proposed method is evaluated in adverse circumstances, where objects in the scene are scarce. It also evaluates the ability of the system to recover from large amounts of scale drift when objects are finally observed.

3. Section 2.7.3 confirms the scale-correcting nature of object observations by showing that the error in scale is inversely proportional to the number of object observations near a keyframe.

4. The method is compared to current top-performing monocular odometry methods in Section 2.7.4, which uses the KITTI datasets online assessment tool. As the method does not rely on a fixed camera-height qualitative results are presented from a video sequence captured with a hand-held camera at varying heights in Section 2.7.7.

5. Section 2.7.5 experimentally confirms the selected object extent for the chosen class of cars which was used throughout testing.

(a)



(b)

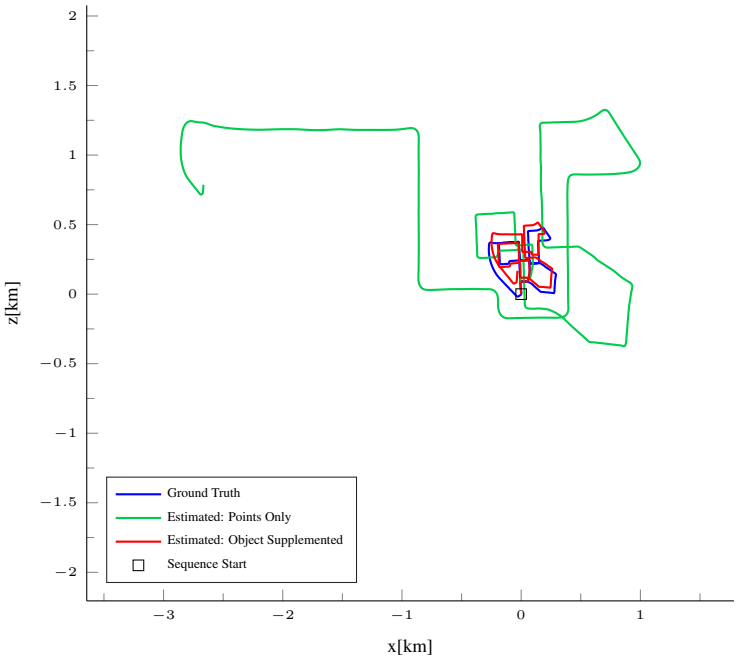
Figure 2.10: Example (a) tracking frame and (b) map generated by the method on a sequence from the KITTI dataset. Localised objects are shown as cubes.

6. Section 2.7.6 outlines how the size distribution and noise model for a particular object class may be acquired.

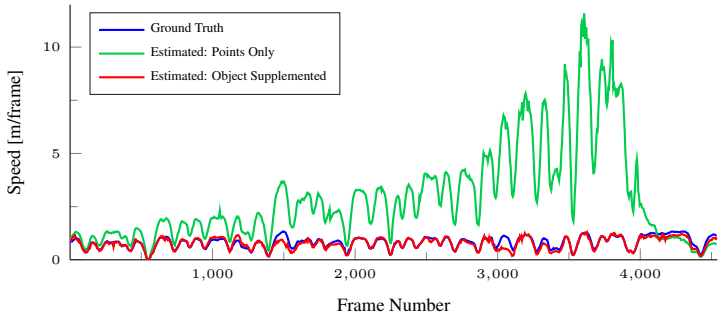
2.7.1 Comparison with bundle adjustment with landmarks alone

To demonstrate the scale-correcting nature of object-supplemented bundle adjustment, the algorithm has been run on a number of video sequences from the KITTI dataset, first using a landmark-only bundle adjustment and then with an object-supplemented one.

Fig. 2.11(a) compares the recovered trajectories and (b) compares camera speeds, a useful proxy for scale for one of sequences from the KITTI dataset. The speed of the camera at keyframe



(a)



(b)

Figure 2.11: Comparison between (a) the trajectories obtained using monocular SLAM with a bundle adjustment using landmarks alone and (b) those obtained using a bundle adjustment supplemented with objects for KITTI sequence 0. Ground truth data is included for comparison.

i was calculated as a backwards difference $s_i = ||\mathbf{c}_i - \mathbf{c}_{i-1}||$, where $\mathbf{c}_i$ is the keyframe i's camera centre in world coordinates. Although both methods experience rotational and translational drift, the object-supplemented method is able to stay well within the actual speed of the camera. The landmark-method accumulates considerable amounts of error due to the scale drift it experiences.

Table 2.1 compares a root mean square error E_{RMSE} for the trajectories recovered with and without objects for the first 11 sequences in the KITTI dataset which come with ground truth data. This is a scale-invariant measure of the translation error for a monocular sequence given by (*c.f.* [75])

$$E_{\text{RMSE}} = \sqrt{\frac{1}{N} \arg\min_{s} \sum_{i \in \mathcal{T}} (\mathbf{t}_i - s\hat{\mathbf{t}}_i)^2} \,. \tag{2.33}$$

Seq #	No of frames	E_{RMSE}, this work [m]	E_{RMSE}, no objects [m]	Seq #	No of frames	E_{RMSE}, this work [m]	E_{RMSE}, no objects [m]
0	4541	**73.4**	1181.0	6	1101	**73.1**	244.9
1	1101	**545.8**	712.0	7	1101	**47.1**	110.2
2	4661	**55.5**	815.7	8	4071	**72.2**	1907.6
3	801	**30.6**	81.1	9	1591	**31.2**	139.6
4	271	10.7	**7.4**	10	1201	**53.5**	115.3
5	2761	**50.8**	798.8				

Table 2.1: RMSE with (this work) and without objects for the first 11 sequences in the KITTI dataset.

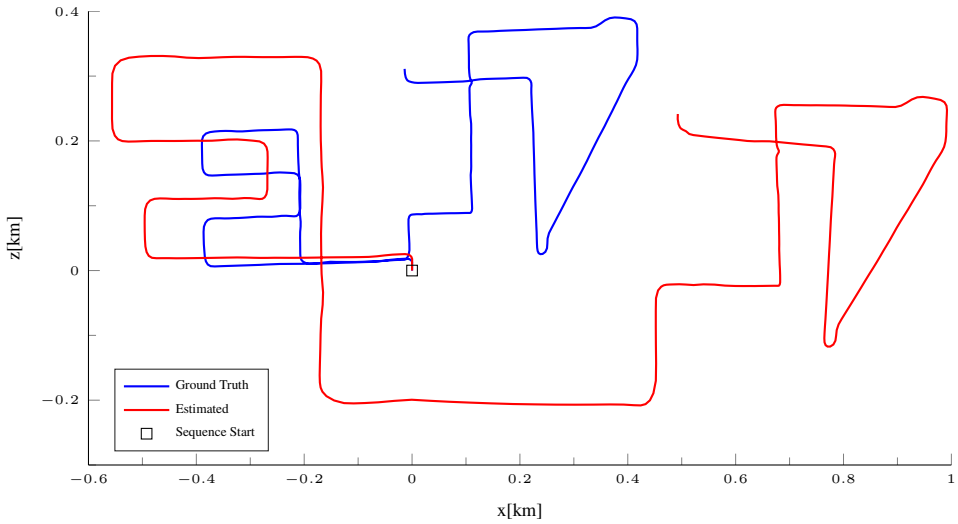
Figure 2.12: Example frames from KITTI sequences 1 (left) and 4 (right). The sequences contain no stationary objects.

where s is a global scale factor applied to the estimated trajectory, and $\mathbf{t}_i$ is the translational component of keyframe i. As the estimated trajectory is metrically accurate, however, Table 2.1 shows values with $s = 1$.

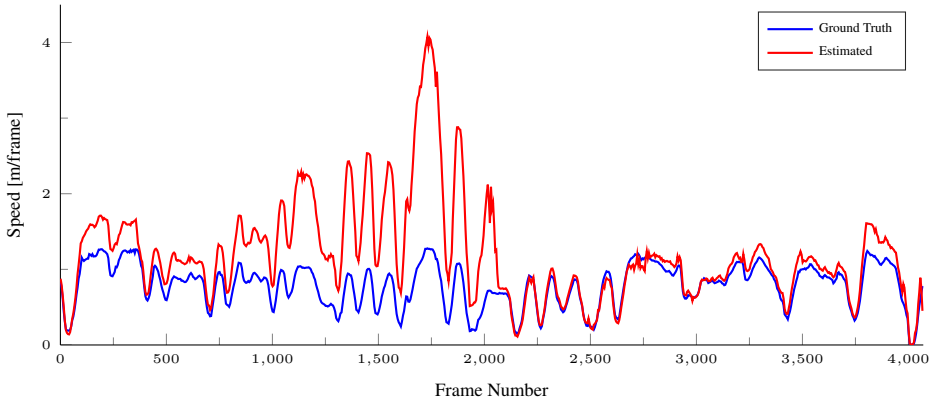
For all but two sequences, the use of objects results in a marked decreased in translation error. It turned out that sequences 1 and 4 continue to have a high amount of error mainly due to the lack of stationary objects in these sequences as shown Fig. 2.12.

2.7.2 Recovery after scale drift

One of the failings of our preliminary work (see §2.4.2)(and that of Castle *et al.* [43]) was that the introduction of scale correction after substantial drift had occurred (say because of a lack of object measurements) was disruptive: in our case causing good measurements to be rejected, and in [43] causing terminal failure. Fig. 2.13 demonstrates that this is not the case here. It shows the estimated trajectory and camera speed obtained by ignoring all object measurements for the first 2000 frames. Almost immediately after objects are used again, the scale is corrected to a value close to ground truth.



(a)



(b)

Figure 2.13: Estimated trajectory (a) and translational speed (b) obtained by initially ignoring objects in the first 2000 frames of the KITTI sequence 8. At frame 2000, once objects are included in the bundle adjustment, scale is promptly corrected.

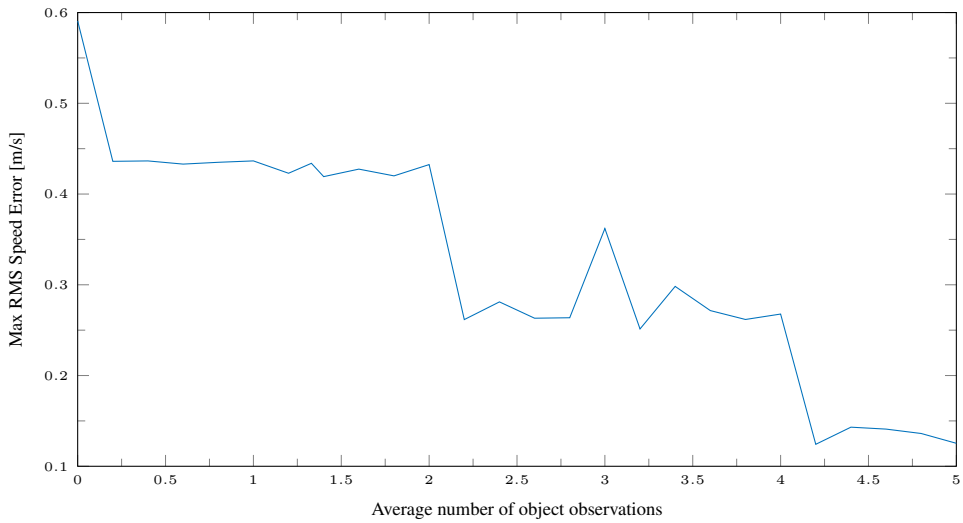


Figure 2.14: Plot of maximum RMS speed error vs neighbourhood-averaged object observations.

2.7.3 Effect of object observations on speed estimation

While Section 2.7.1 shows that there is a clear advantage of using object-supplemented bundle adjustment over its landmark-only counterpart, this experiment presents a more localised view of how objects affect scale estimation. For each keyframe in an estimated trajectory the RMS error in speed is computed, along with the average number of object observations that are present in a 10-keyframe neighbourhood. The number of object observations are averaged spatially as they generally affect a number of surrounding keyframes rather than a single one. A single keyframe with no object observations surrounded by keyframes with observations is still likely to have a low speed error despite having no observations itself. Keyframes are then grouped according to their average number of observations, and the maximum RMS error in speed experienced by this group is recorded. Fig. 2.14 shows a plot of the maximum error vs the average number of observations for KITTI sequences 0 to 10. Sequences 0 and 4 are excluded due to their lack of any stationary objects. The figure shows that increasing the number of observations does indeed place an upper bound on the error that is experienced.

2.7.4 An (unfair) online assessment

The results for the last 10 sequences of the KITTI dataset have been evaluated using the online assessment tool at *http://www.cvlibs.net/datasets/kitti/eval_odometry.php* [76, 72] for comparison with other researchers' results. The dataset uses error metrics for rotational and translational error.

$$E_{rot}(\mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} \angle[(\hat{\mathtt{T}}_j \ominus \hat{\mathtt{T}}_i) \ominus (\mathtt{T}_j \ominus \mathtt{T}_i)] \tag{2.34}$$

$$E_{trans}(\mathcal{T}) = \frac{1}{|\mathcal{T}|} \sum_{(i,j)\in\mathcal{T}} ||(\hat{\mathtt{T}}_j \ominus \hat{\mathtt{T}}_i) \ominus (\mathtt{T}_j \ominus \mathtt{T}_i)||_2 \ , \tag{2.35}$$

where $\mathcal{T}$ and $|\mathcal{T}|$ is a set of frames (i, j) and its length respectively, $\hat{\mathtt{T}}$ and $\mathtt{T}$ are estimated and ground-truth poses respectively, $\ominus$ is the inverse compositional operator [77], and $\angle[\,]$ is the rotation angle.

The average performance over all monocular SLAM systems is captured by some 9% translation error and 0.020 deg/m rotational error, while the best performance [52, 53] achieves 2.4% translation error and 0.006 deg/m rotational error. Unfortunately for comparison purposes, all ref-

erence methods employ a planar constraint in one form or other. The proposed method yields 20% translation error and 0.014 deg/m rotational error on average, results which at least approach, and for rotation exceed, these other methods.

A number of possibilities are considered for reducing error. First, it is likely the large inter-frame motion in the KITTI sequences is over-stretching the feature and camera tracking stage of PTAM, which was developed specifically for small AR workspaces. A simple KLT-based [78] odometry system is able to find numerous matches between every pair of frames on sequences where PTAM lost a number of map points. Sequence 1 in Table. 2.1 is one such example.

Secondly, a number of the test trajectories do not contain many static objects of the selected interest class, making scale drift correction impossible on these sequences. A prime example of this is sequence 14, which takes place in a park where the only scenery are a number of rows of shrubs. The need for variety in object classes is discussed later.

A third potential source of error is that of setting the object extent parameter at a value slightly larger or slightly smaller than reality. This possibility is explored next.

2.7.5 Object extent I: selection

As mentioned earlier, the object extent ϵ for detected cars was set as an average from the ranges of popular make. However, an empirical search over this parameter is also possible. To select an optimal extent parameter enumerate a range of plausible values and evaluate the performance of the system on all training sequences of the KITTI dataset that contain cars. The average per-frame RMS error in estimated speed is then

$$E_{speed} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(s_i - \hat{s}_i\right)^2}\,, \tag{2.36}$$

where the speed s_i is computed using the backward difference in camera centres discussed in Section 2.7.1 and N is the total number of estimated keyframes. The error is shown for varying object radius/extent values in Fig. 2.15. The value with the lowest error coincides with the choice of the object extent $1.2\,\text{m}$.

2.7.6 Object extent II: learning the distribution

Our results to this point have considered all objects from a particular class to have a constant extent. This section details how a distribution over this parameter may be estimated from a video sequence where a ground-truth pose for each from is available. The method re-frames the problem so to speak; rather than allowing prior information on object extent to determine the speed of the camera and therefore poses of keyframes, known correct poses of frames are used to estimate the extent for objects.

Assuming a set of ground truth keyframe poses $\mathcal{T}_{GT}$ with associated object observations, a set of objects $\mathcal{Q}$ of the particular class in question is estimated that minimises

$$\mathcal{Q} = \arg\min_{\mathcal{Q}} \sum_{j \in \mathcal{T}_{GT}} \sum_{i \in \mathcal{Q}} \mathbf{r}_{ki}^{\top} \mathbf{r}_{ki} \ , \tag{2.37}$$

which is simply the object term in Eq. (2.22) without associated covariance matrix. Note, however, that the extent ϵ of each object is no longer a constant hyper-parameter for the entire class, but is a free parameter estimated for *each* object alongside its position. Fig. 2.16 shows the resultant distribution when estimating objects for KITTI sequences 0 to 10. The distribution is approximately Gaussian, with mean 1.2 and variance 0.2, which is in agreement with the earlier assumptions.

The distributions for errors associated with objects may also be estimated using this method by examining residuals after optimisation has converged. Fig. 2.17 shows the final errors in the centre components of object measurements. The error is approximately Gaussian and zero-mean as expected, with the x-component having a variance of 43.4 and the y-component having a variance of 16.8. The boundary component of object errors do slightly break the Gaussian

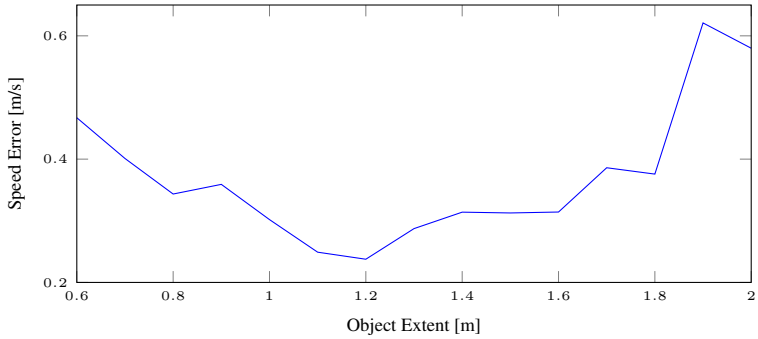


Figure 2.15: Per-frame average RMS error in speed versus object extent for KITTI training sequences that contain cars.

assumption as shown in Fig. 2.17. The means of the errors in width and height are 10.4 and -11.6 respectively. This is almost certainly due to some steady-state error in some rectangular measurements. Here spherical objects are unable to completely fit detections correctly. Fig 2.19 is one such example. The covariance of boundary errors is

$$\mathrm{Cov}(E_w, E_h) = \begin{bmatrix} 190.0 & -123.4 \\ -123.4 & 128.2 \end{bmatrix}, \tag{2.38}$$

which indicates the errors are negatively correlated. This is expected considering the rectangular detection problem, as rectangular detections are likely to keep their aspect ratio regardless of their estimated scale.

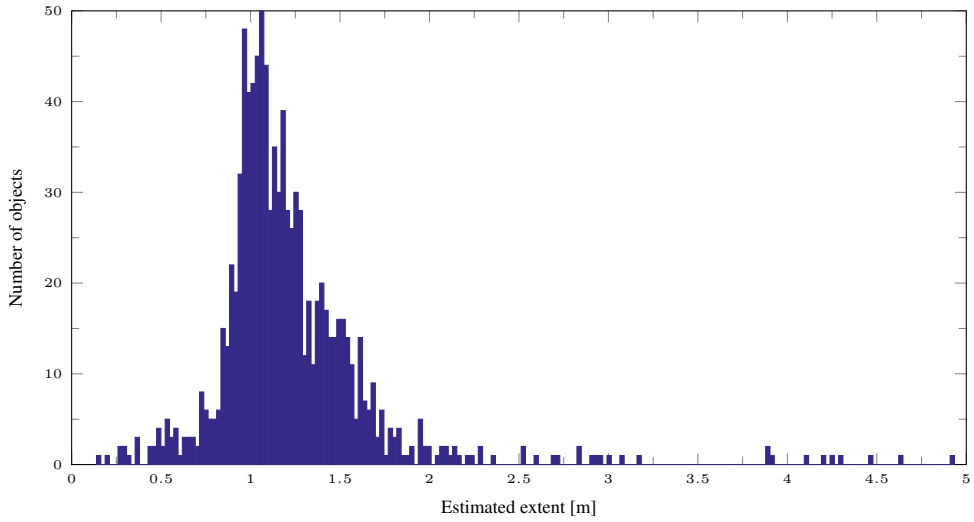


Figure 2.16: Frequency of estimated object extent for KITTI dataset.

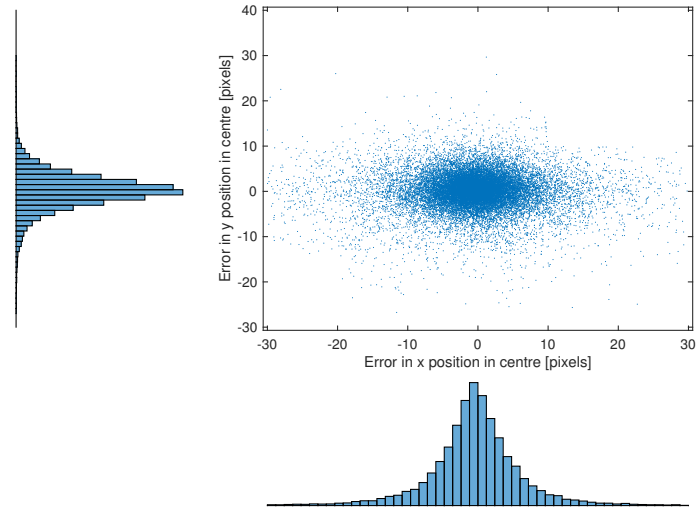


Figure 2.17: Scatter plot of error in centre component of object detections.

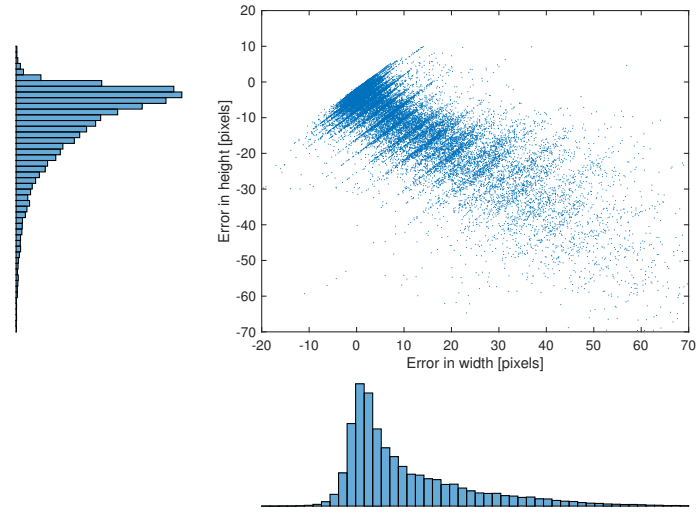


Figure 2.18: Scatter plot of error in object bounding box width and height.

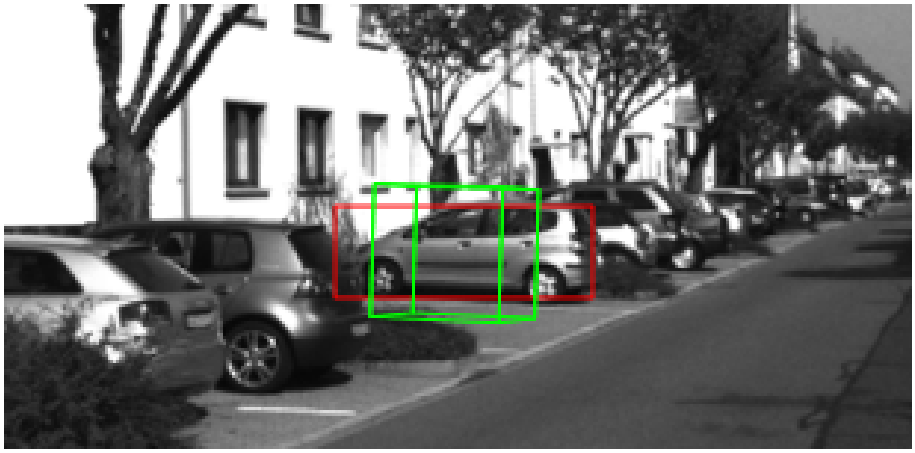


Figure 2.19: Object unable to correctly fit rectangular detection.

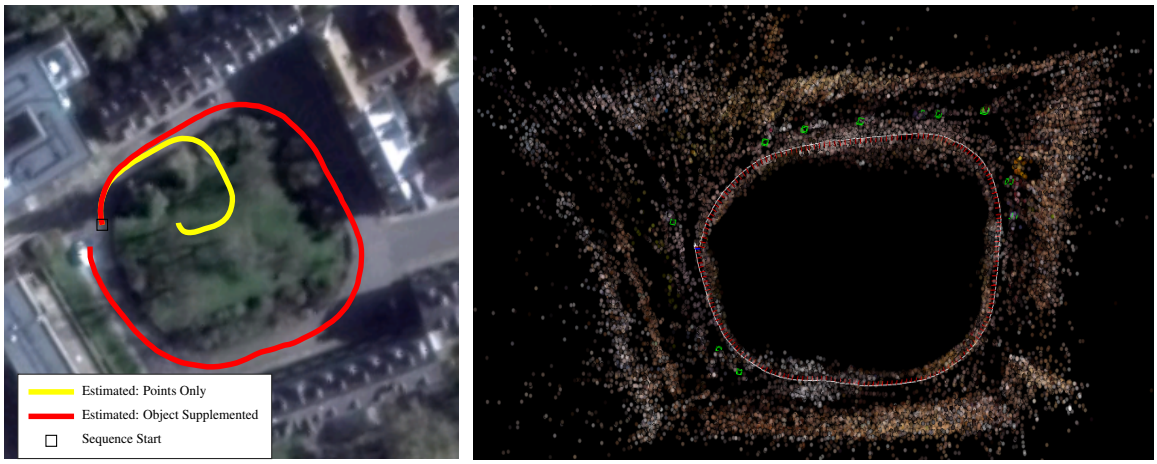
2.7.7 Sequence from a hand-held: varying camera height

We suggested that the comparison against the state of the art provided by the KITTI online assessment tool was not a fair test of the current method because the best performers all constrain their cameras to have fixed height about the roadway.

To demonstrate the ability of our method to function without such a constraint an outdoor sequence was captured using a hand-held phone camera (a Samsung Galaxy S6, pre-calibrated with fixed focal length). As ground truth was not available, the sequence contained a small loop, and performance was evaluated on the qualitative distance between the start and end of the loop. Fig. 2.20 shows sample images from the sequence with the camera at different heights about the ground plane.

Fig. 2.21 shows the resultant trajectories estimated using a landmark-only and an object-supplemented bundle adjustment, respectively, overlaid onto satellite imagery of the actual area where the sequence was filmed. Cars with an extent of $\epsilon = 1.2\,\mathrm{m}$ were once again used as the class and extent for the object-supplemented method. Comparison of the trajectories with satellite imagery shows that object-supplemented bundle adjustment provides a marked decrease in scale drift over its landmark-only counterpart. A detailed point cloud of the mapped area is shown in Fig. 2.21.

Figure 2.20: Images from video sequence with varying height. Sequence with a high camera height (left) and a low camera height (right).



(a) (b)

Figure 2.21: (a) Resultant trajectories estimated using point-only bundle adjustment (yellow), and object-supplemented bundle adjustment (red). Satellite imagery of the mapped area is shown for comparison. Satellite imagery: Google, DigitalGlobe. (b) Point cloud from video sequence with varying height. Objects are shown in green.

2.8 Conclusion

A new method has been presented that is able to incorporate scale information from object classes into vision-only monocular SLAM systems without significant increases in computational complexity.

It has been shown that the method is able to maintain a correct scale estimate throughout a trajectory containing enough known object classes. Even when object classes have not been observed after a significant amount of time, the resultant scale drift after classes are seen again. While other methods rely on external hardware or assumptions about the height of the camera above the ground plane, the proposed method only assumes that there is some known object in the

map. This is ideal for general purpose applications, when other assumptions no longer hold.

A number of avenues might be pursued to improve performance. The method currently isn't able to correct scale in parts of the map outside of the local bundle adjustment window. In Section 2.7.2 for example, the bundle adjustment isn't able to correct parts of the trajectory where no object observations are available.

Recent works [79], however, have shown that a global bundle adjustment is unnecessary as accurate structure is only required around the current camera estimate. Rather, they apply a joint "double-window" optimisation that refines structure and keyframes in a local adjustment window, and a pose-graph optimisation to the rest of the map. One possible avenue of research is the investigation of whether scale information from a local object-bundle adjustment can be propagated correctly along the pose-graph using similarity transformations [55] in this formulation. This additionally would allow for simple closure of loops.

One challenge faced on the KITTI dataset was a lack of sequences with static objects of the chosen object class. Due to the simplicity of the method, adding object classes is trivial only requiring a viable object detector and tracker. Care has to be taken to correctly balance possibly conflicting scale information from different object classes. Sufficient sampling of each object class's size using the method proposed in Section 2.7.6 would be necessary.

If metric accuracy isn't a concern, online estimation of object size-distribution is also possible. This would require estimating the distribution in areas with little drift and then utilising the distribution in later parts of the map.

Currently object data association between frames is performed independently prior to running the method. Performing this online is quite possible by utilising a separate thread dedicated to detection and association. A possible extension of this would be to

system, however the effects of drift on tracking performance need to be evaluated. Moving objects in the map pose a problem to any SLAM method and better rejection of moving objects would be beneficial.

Chapter 3

Scale drift correction using monocular depth priors

In this chapter a deep convolutional neural network is used to predict depth maps from monocular images, which provide priors on the depths of landmarks in the image. Priors are then used as a regulariser in a modified bundle adjustment to correct scale drift. The method is tested on a long range dataset and obtains results that are scale drift-free.

3.1 Introduction

The previous chapter described a method for correcting scale drift using object classes with a known size-distribution. The method was able to correct scale drift over long trajectories, but suffered from a few shortcomings. Apart from intra-class size variation and the presence of moving objects leading to inaccurate estimation, the most obvious and important difficulty was that instances of the known object class or classes have to be present in the trajectory.

Even when objects from known classes are observed and well behaved, scale recovery is still dependent on a low number of measurements. For example, even in the KITTI database of street scenes, the *maximum* number of cars in any one frame was a mere six.

Recent progress in machine learning suggests a different approach. Rather than relying on a sparse number of measurements to hint at size and scene depth, in this chapter we propose augmenting landmark measurements with depth priors delivered by a deep convolutional network (CNN) which is trained on sequences for which absolute depth is known. A number of works have shown the possibility of estimating depth monocularly over the *entire* image. Although the

requirement is for depth only at the usual number of relatively sparse landmarks in our case, we employ one of these networks to provide depth estimates on landmarks in the image.

First, in Section 3.3, we make the changes to bundle adjustment to admit priors and test the modifications using those KITTI sequences for which ground-truth depth is available from LiDAR. Next, in Section 3.4, a convolutional neural network trained using image structure as input and a depth image as output is employed. In Section 3.5, the CNN is used to obtain priors on the landmarks' scene depths, and the scene structure and cameras are optimised using the bundle adjustment with priors. Finally, concluding remarks and directions for future research are presented in Section 3.6.

Before the main contribution is presented, Section 3.2 reviews methods for depth estimation from a single image.

3.2 Related work

Depth estimation from a single image is an ill-posed problem; even ignoring scale ambiguity inherent in a monocular view, an infinite number of scenes may all give rise to the same single image. By ignoring physically improbably scenes, however, more probable one may be accurately estimated up to scale.

In pioneering work Michels *et al.* [80] used absolute depth data along 1D strips from a LiDAR to learn about pairwise relationships between pixels in that strip. This information was used to control an autonomous vehicle. Saxena *et al.* [81] learned depth maps for an entire image, utilising an MRF to estimate depths and to model relationships between them in an image. The work uses two types of small convolutional features; one for estimating absolute depth of a single patch, and one for estimating the relative distance between a pair of patches. The calculations occur at multiple scales to account for changes in size with depth and to allow more contextual information to be utilised. While the method attempts to model absolute depth, results show that the network produces far more erroneous results when predicting these versus relative depth estimates. The authors put this down to lack of training data. The depth information from this method is used for full 3D reconstruction in [82].

More recent works have greatly increased accuracy by utilising deep networks trained end to end. Eigen *et al.* [83] utilise a pair of deep networks in a coarse to fine manner. An initial network

globally predicts the entire depth map for a single image on a coarse level, the result of which is locally refined using a fine-scale network. The work shows promising results for both absolute and relative depth estimates. The outputs of a CNN can be further refined by utilising a conditional random field (CRF) that models relationships between pixels [84, 85], further increasing accuracy.

In this chapter, we attempt to use the depths inferred from a single image as priors in monocular SLAM.

3.3 Bundle adjustment with depth priors

We first detail the method of incorporating depth priors into a bundle adjustment. Recall that standard bundle adjustment finds the set of landmarks in the world frame w, $\mathcal{X} = \{\mathbf{X}_{0w}, \mathbf{X}_{1w}..., \mathbf{X}_{Jw}\}$ and keyframe poses $\mathcal{T} = \{\mathtt{T}_0, \mathtt{T}_1, ..., \mathtt{T}_M\}$ that minimise the reprojection error:

$$\{\mathcal{X}, \mathcal{T}\} = \underset{\{\mathcal{P},\mathcal{T}\}}{\arg\min} \sum_{i\in\mathcal{X}} \sum_{j\in\mathcal{T}} \mathbf{r}_{ji}^{\top} \mathtt{W}_{ij}^{-1} \mathbf{r}_{ji} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min} E_{\text{reproj}} \ , \tag{3.1}$$

where $\mathbf{r}_{ji}$ is the residual in the image.

Now assume the absolute depth of the landmark $\mathbf{X}_{iw}$ may be determined as a prior in a particular keyframe j as ρ_{ij} and that the prior is normally distributed as

$$\hat{\rho}_{ij} \sim \mathcal{N}(\rho_{ij}, \sigma_{ij}^2) \ . \tag{3.2}$$

This information can be incorporated into the adjustment by regularising the reprojection cost with a term

$$E_{\text{depth}} = \sum_{i\in\mathcal{X}} \sum_{j\in\mathcal{T}} \frac{1}{\sigma_{ij}^2} (Z_{ij} - \hat{\rho}_{ij})^2 \ , \tag{3.3}$$

that penalises deviation from the priors. Z_{ij} is the landmark's z-coordinate in the coordinate system defined by $\mathtt{T}_j$. The modified bundle adjustment is

$$\{\mathcal{X}, \mathcal{T}\} = \underset{\{\mathcal{P},\mathcal{T}\}}{\arg\min} E_{\text{reproj}} + \lambda E_{\text{depth}} \ , \tag{3.4}$$

where λ is a constant used to adjust weighting between the two errors. For tests throughout this chapter we will use a value of 1, although this may be adjusted depending on the certainty of the source of priors.

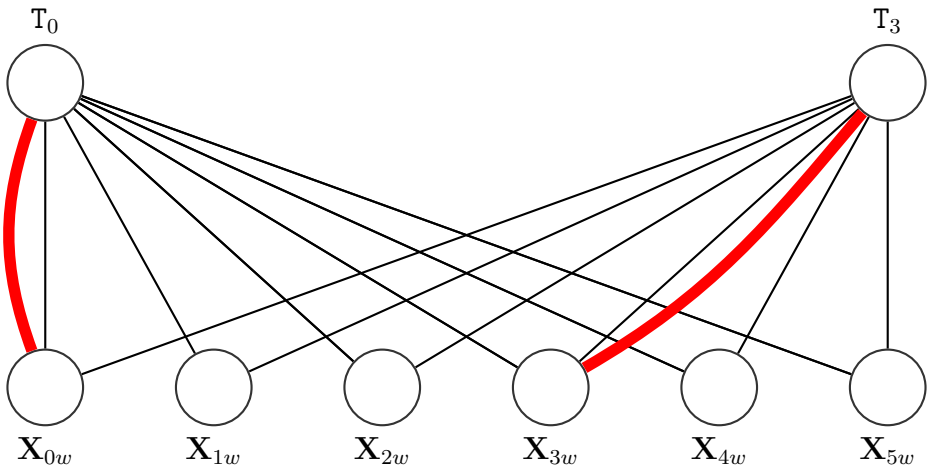


Figure 3.1: MRF representation of bundle adjustment with depth priors. Scale-dependent depth prior constraints are shown in red. Rather than introducing additional parameters that need to be jointly optimised, the proposed solution only adds new measurements.

The MRF of the resultant bundle adjustment is shown in Fig 3.1. Rather than adding additional parameters that themselves require estimation, the depth prior bundle adjustment simply adds additional scale-dependent measurements between landmarks and keyframes as shown in red.

Minimisation of the cost containing E_{depth} requires the derivative of Z_{ij} with respect to $\mathbf{X}_{iw}$ and $\mathtt{T}_j$. The derivative of the landmark in camera coordinates with respect to its position in world coordinates is

$$\frac{\partial \mathbf{X}_{ij}}{\partial \mathbf{X}_{iw}} = \mathtt{R}_j \ , \tag{3.5}$$

where $\mathtt{R}_j$ is the top-left 3×3 rotation matrix of the euclidean transform $\mathtt{T}_j$. Likewise, for a small change to $\mathtt{T}_j$ through the lie algebra $\boldsymbol{\mu} \in \mathfrak{se}(3)$

$$\frac{\partial \mathbf{X}_{ij}}{\partial \mu_k} = \mathtt{G}_k \begin{bmatrix} \mathbf{X}_{iw} \\ 1 \end{bmatrix} \ , \tag{3.6}$$

where μ_k is the kth element of $\boldsymbol{\mu}$, and $\mathtt{G}_k$ is the kth generator matrix of the special euclidean group $\mathbf{SE}(3)$. The derivatives $\frac{\partial Z_{ij}}{\partial \mathbf{X}_{iw}}$ and $\frac{\partial Z_{ij}}{\partial \mu_k}$ are simply the third row/element of $\frac{\partial \mathbf{X}_{ij}}{\partial \mathbf{X}_{iw}}$ and $\frac{\partial \mathbf{X}_{ij}}{\partial \mu_k}$ respectively.

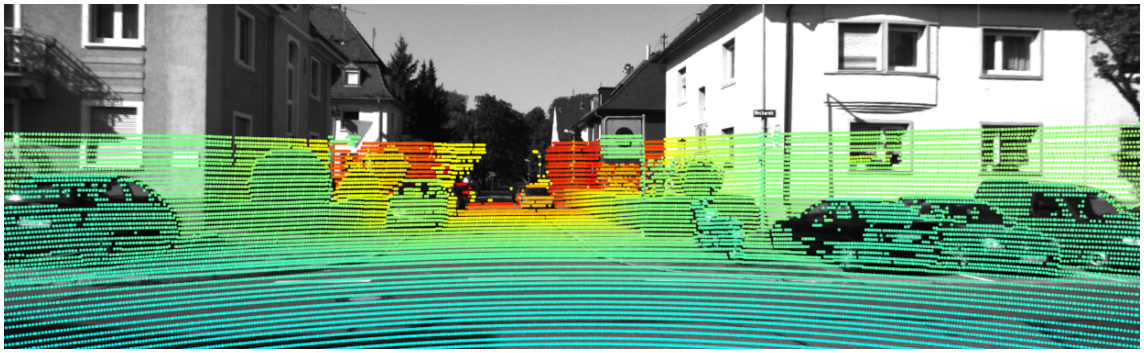


Figure 3.2: LiDAR point cloud from KITTI dataset re-projected into its associated camera image. False colours indicate depth from camera.

3.3.1 Testing the adjustment using priors from LiDAR data

The implementation and operation of the modified bundle adjustment have been verified using highly accurate prior information obtained form the KITTI datasets [72, 50] LiDAR point clouds, captured at the same time as the visual images by a Velodyne sensor.

Assigning the prior. A 3D point from the LiDAR is converted to camera-centric coordinates via

$$\begin{bmatrix} \mathbf{X}_{vj} \\ 1 \end{bmatrix} = \mathtt{T}_v \begin{bmatrix} \mathbf{X}_v \\ 1 \end{bmatrix} , \tag{3.7}$$

where $\mathbf{X}_v$ is a point within the velodyne's coordinate system, X_{vj} is that same point within the camera's coordinate system, and $\mathtt{T}_v$ is the transformation which is readily pre-computed using the transformations between sensors supplied with the dataset. Fig. 3.2 shows an example video frame with its corresponding LiDAR point cloud projected into the frame. For these test purposes, there is little value in interpolating depths, and landmark $\mathbf{X}_{ij}$ with measured image position $\mathbf{x}_{ji}$ is given the prior $\hat{\rho}_{ij} = Z_{vj}$, where the LiDAR point $\mathbf{X}_{vj}$ is that which minimises the distance

$$d = \sqrt{\left(\mathbf{x}_{ji} - \mathrm{proj}_K(\mathbf{X}_{vj})\right)^2} , \tag{3.8}$$

provided $d < 5$ pixels.

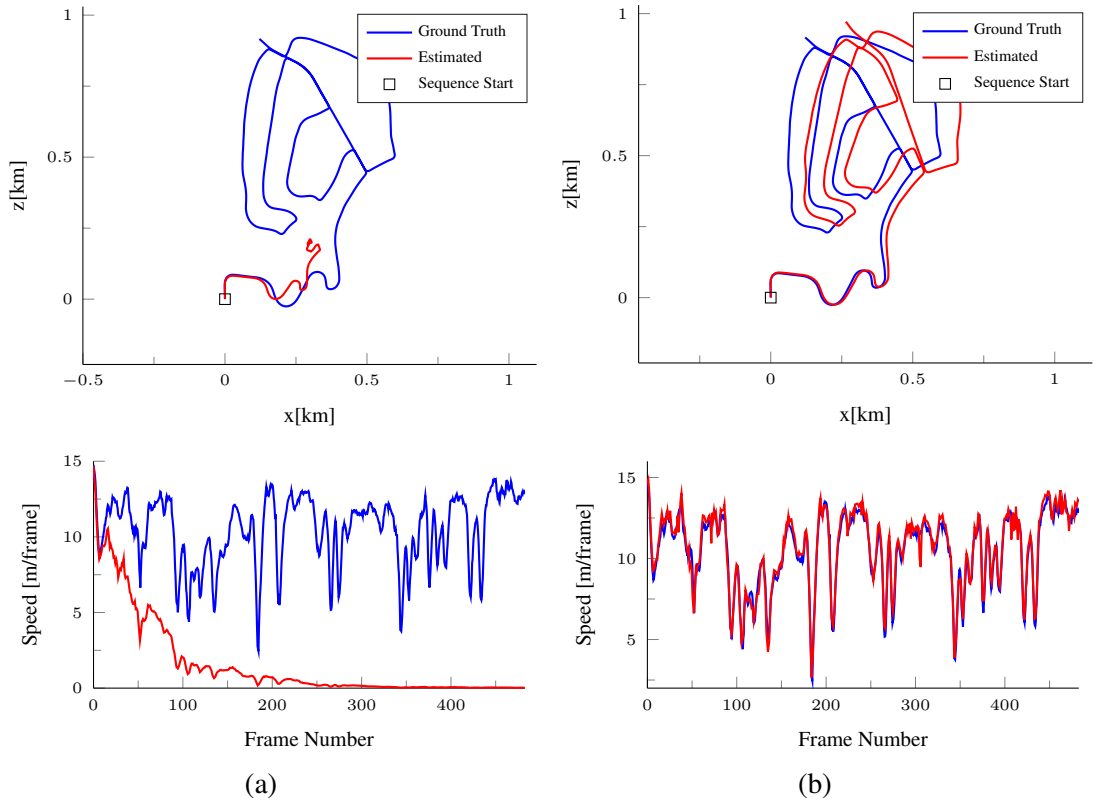


Figure 3.3: Estimated trajectory (top) and speed (bottom) for KITTI sequence 2 using method (a) without and (b) with depth priors.

3.3.2 Results: verification of adjustment with priors

Tests were conducted on four difficult sequences from the KITTI dataset, including sequence 04 which, owing to a shortage of stationary objects, was particularly problematic for the object supplemented bundle adjustment in the previous chapter. A local bundle adjustment *with* and *without* priors were tested on the sequence, using an active window of 5 keyframes.

Fig. 3.3 shows the estimated trajectory and speeds for sequence 02. The system without depth priors on the left slowly drifts in speed until it is almost zero, while the system with depth priors has little trouble maintaining the correct speed throughout the trajectory. Table 3.1 shows the error in speed and error in translation for all test sequences with and without depth priors. For all sequences, including depth priors greatly reduces both errors.

		RMS Error Speed [m/frame]	
Seq. #	Frames	No Priors	With Priors
0	4541	4.3	**0.3**
2	4661	9.1	**0.3**
4	271	2.5	**0.2**
8	4071	5.5	**0.4**

(a)

		RMS Error Trans [m]	
Seq. #	Frames	No Priors	With Priors
0	4541	133.9	**11.9**
2	4661	427.5	**76.2**
4	271	32.8	**3.0**
8	4071	273.4	**28.0**

(b)

Table 3.1: RMS error in (a) speed and (b) translation for adjustments without and with depth priors for selected sequences. The use of priors greatly decreases errors in both. Note that errors for the method without priors differ from those obtained without objects in Table 2.1 due to a difference in base SLAM system. Scale drift tended to decrease speed to zero, while the system used in Chapter 2 tended to increase speeds, leading to much larger distances between estimated and ground-truth camera positions.

3.4 Learning priors from monocular images

With the use of priors validated, the aim now is to avoid using LiDAR data directly, instead using a convolutional neural network to provide priors from a single keyframe's image.

Rather the (re)inventing a new CNN, we use that developed by Eigen *et al.* [83]. They use deep networks trained on both the NYU dataset [86] and on raw data from the KITTI dataset [72]. While the KITTI-trained network would be more suitable to the target domain, it is not publicly available and so the NYU-trained network was used.

3.4.1 Structure of the CNN

The provided network requires a 3 channel input image of 320×240 pixels and produces an output depth map of 74×55 pixels. Input and output images are simply re-sized from and to the KITTI image size (most are around 1241×376 pixels) to meet these requirements. Though fully described in [83], for completeness we will give an overview of its structure here.

Fig. 3.4 gives an overview of the architecture of the network. An initial network first computes the coarse depth using a set of five feature extraction layers each consisting of convolution and max pooling layers. Features are fed to a fully connected layer, which itself is fully connected to the output image. A fine-scale network first performs a set of convolutions and max pooling to the input image. The output of this is a 63-channel feature image, with which the output of the coarse network is concatenated. Two final convolution layers are applied to this new 64-channel image to produce the final depth image.

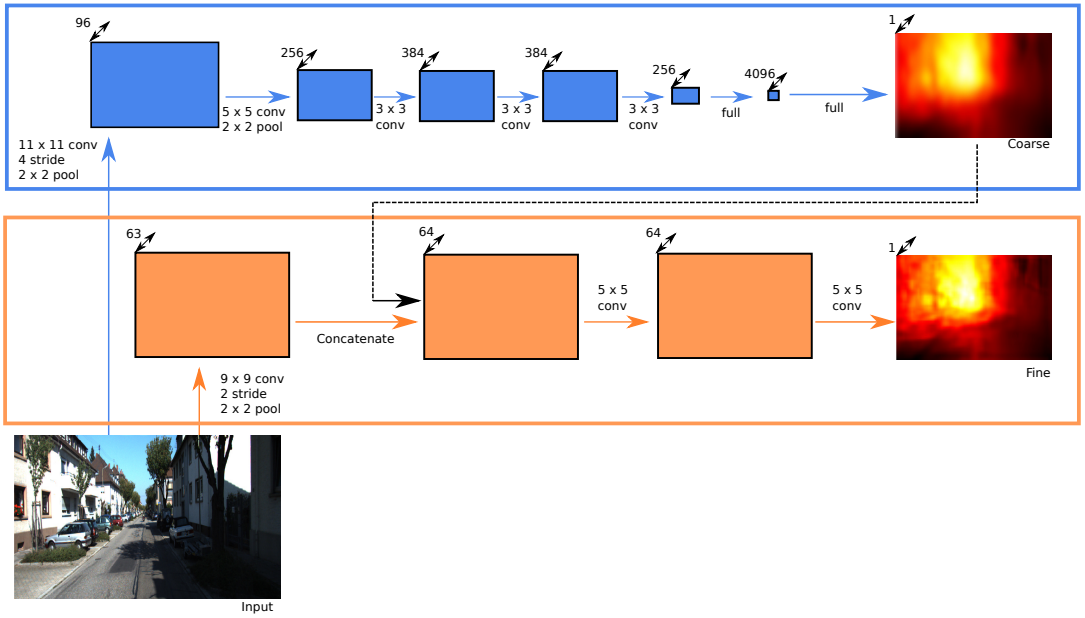


Figure 3.4: Network architecture for depth estimation. Adapted from [83].

3.4.2 Results from the CNN

Fig 3.5 shows a typical result, with (a) input from the KITTI odometry dataset and (b) subsequent depth estimation output from the CNN.

As one would expect of any monocular depth estimation algorithm, the CNN is better at predicting *relative* depths between pixels rather than predicting absolute depth at any one individual pixel. This is made obvious by comparing CNN depth output with ground-truth provided by LiDAR. The scale factor

$$s = \frac{d_{\text{LiDAR}}}{d_{\text{CNN}}}\,, \tag{3.9}$$

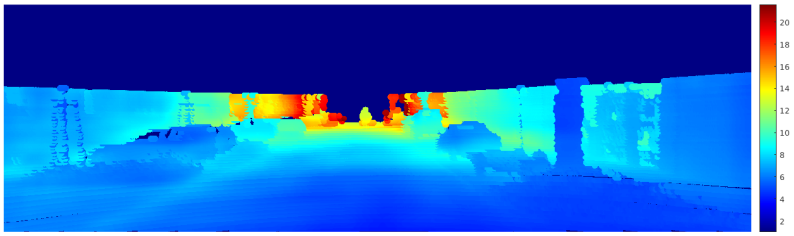
should be unity everywhere, but as shown in Fig 3.5(c), this is not the case. The upper parts of this image should be ignored as LiDAR measurements are available only in approximately the lower half (as clear from Fig. 3.2). In the lower part the scale factor is almost constant across most pixels indicating good relative depth estimation, but is larger than unity. The CNN's underestimation is, not surprisingly, more pronounced at greater depth.

3.4.3 Correcting the CNN output

The most likely reason for underestimation of depth is the difference between application and testing domains of the network. Originally trained on the NYU dataset, the network understandably

(a)

(b)



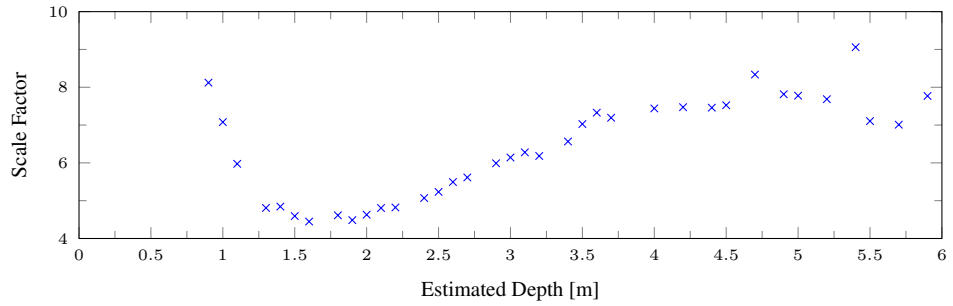
(c)

Figure 3.5: Example (a) input image and (b) output depth map from the CNN of [83]. In (b) the scale of the false-colour images is in metres. (c) The scale factor $d_{\mathrm{LiDAR}}/d_{\mathrm{CNN}}$ across the lower part of the output where LiDAR is available. A non-unit scale factor indicates incorrect depth estimation. For much of the image the underestimation of the CNN is consistent at a factor around 5.

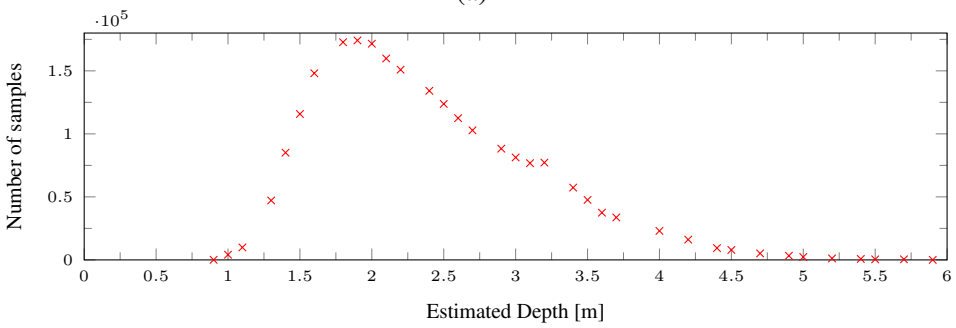
struggles with the outdoor domain of the KITTI dataset. Regardless, its relative depth estimation is accurate for closer depths, and therefore we proceed to correct its output to ensure better estimation of absolute depth.

As LiDAR data is available for much of the dataset, we now use it to determine an *overall* relationship between the CNN estimate and ground truth. (Whether such overall corrections can be found for other domains is an open question.)

Fig 3.6(a) shows the average scale factor v.s. discretised depth from a sample of about 3 million pixels. Pixels were sampled uniformly from random images in the dataset. Fig 3.6(b) shows the number of samples for each depth.



(a)



(b)

Figure 3.6: (a) Scale factor and (b) number of samples vs estimated depth for random sample of depth points.

We choose to correct the output of the CNN using the following strategy. As there are a larger number of samples from the depths between 1.25 and 2.5, we only accept these estimated depths as priors. For this window, the scale factor of 5 is used to correct the output of the CNN. As LiDAR data is only available for the lower half of the image, valid depth estimates from the upper parts of the image are excluded.

3.5 Using adjusted CNN priors in bundle adjustment

3.5.1 Implementation details

We now move to utilising the proposed CNN to provide depth priors in a monocular SLAM method. An overview of the method is shown in Fig. 3.7. As the bundle adjustment and CNN subsections have already been discussed, we now outline the tracking and keyframe-addition subsections. Although similar to those in Chapter 2, they differ slightly in their operation due to a novel method of data association. Both are discussed below.

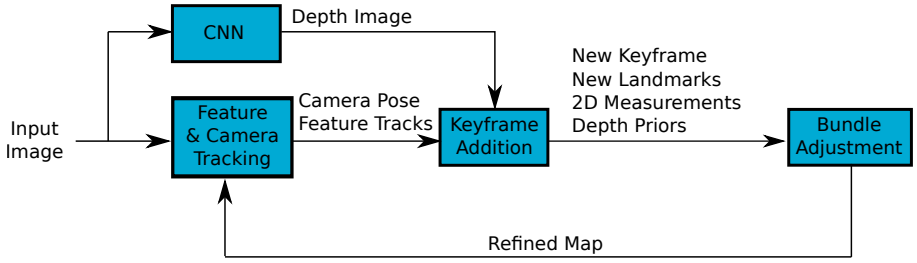


Figure 3.7: Overview of the proposed algorithm.

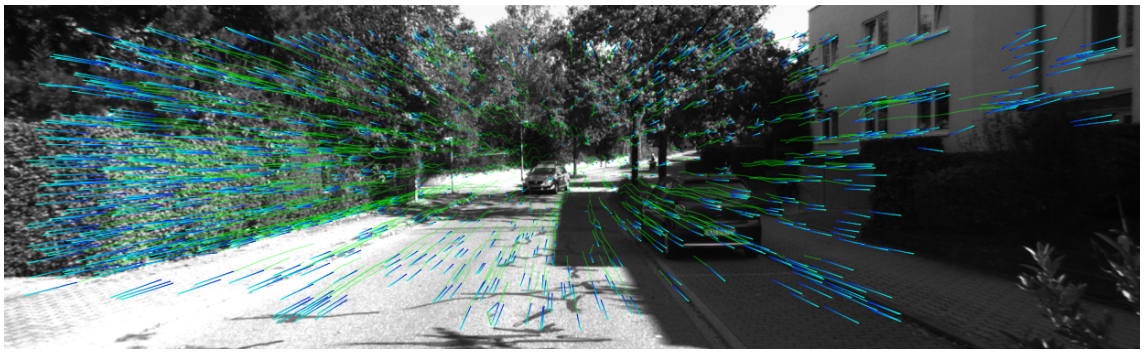


Figure 3.8: Data association using feature tracks. Blue lines indicate matches with the previous frame, which are added to persistent feature tracks indicated by green traces.

Feature and camera tracking. In the previous chapter, PTAM [9] was used as a base SLAM algorithm for the proposed method. For outdoor datasets such as the KITTI dataset, a particular problem was that of data association, as PTAM struggled to find landmarks in new images due to large camera displacements. As a solution to this, we introduce a more robust method of solving data association.

Rather than searching for landmarks at each frame, data association is solved by maintaining a set of feature tracks at each image. The libviso2 visual odometry system [49] is used as a basis due to its availability and speed. For each frame, it produces a set of matches with the previous one. Rather than using this to estimate relative pose estimates as is done in [49], these matches are added to tracks which consist of a list of past 2D image locations. When tracks are lost, new ones are initialised to keep the number of tracks constant. Fig. 3.8 shows an example of matches and tracks used in the method.

When a landmark is initialised, it stores a unique track identifier with it. Solving data association for tracking is simply a matter of finding which of the current feature tracks have an associated landmark.

Keyframe addition. When a new keyframe is added, its image is used as input to the CNN, which outputs the estimated depth image. Image measurements for the keyframe are obtained from the most recent 2D image locations of each of the current feature tracks. For each measurement, a depth prior is obtained by searching in the depth image at its 2D location. This depth is then corrected using the method proposed in Section 3.4.3.

3.5.2 Experiment and results

The bundle adjustment with priors is now used again, but using priors from the CNN, rather than those from LiDAR.

The system is tested again using the depth prediction method proposed in the previous section. Fig. 3.9 shows the estimated trajectory and speed for sequence 00. While scale drift is corrected, the estimated trajectory is not metrically accurate. This is very likely due to the fact that the CNN excels at determining relative depths, but fairs poorly in estimating absolute ones. As previously discussed, this is probably due to a difference between CNN training and test domains.

As estimated trajectories are not metrically accurate, we re-scale them such that the distance between the first two keyframes of each trajectory is equal to that of their respective ground-truth poses. Fig. 3.10 shows the estimated trajectory and speed for sequence 02, where estimated speed (although up to scale) does match ground-truth speed in most cases.

Table 3.2 shows the error in estimated speed and translation for the test sequences. Errors for trajectories estimated without priors and when using priors from LiDAR data are included for comparison. Although slightly less accurate than LiDAR data, the use of a CNN greatly reduces error compared to the trajectories without it. Sequence 4 is an exception. This is potentially due to the nature of the sequence which was predominated by very large depths.

3.6 Conclusion

In this chapter a novel method for incorporating prior knowledge of scene depth into monocular bundle adjustment has been presented. The bundle adjustment was tested on a selection of sequences from the KITTI dataset using a newer data association method that maintains a constant number of feature tracks rather than searching for landmarks in each frame.

First, priors were obtained from ground-truth depth provided by LiDAR data. Using this data

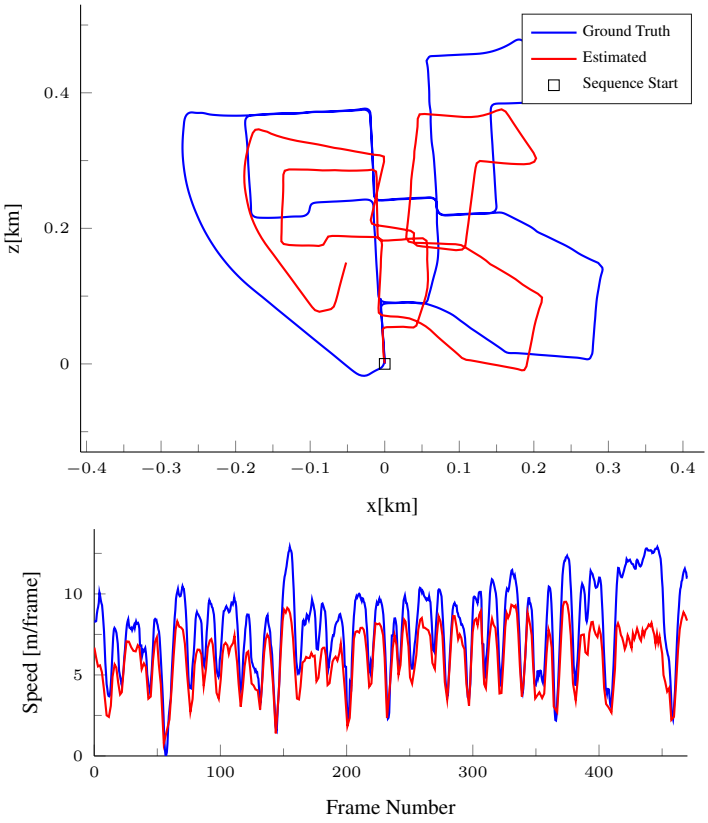


Figure 3.9: Estimated trajectory (top) and speed (bottom) when using the system with CNN-derived depth priors on KITTI sequence 00. While scale drift is controlled, the sequence is only accurate up to scale.

	RMS Error Speed [m/frame]		
Seq. #	No Priors	CNN	LiDAR
0	4.3	0.9	**0.3**
2	9.1	1.3	**0.3**
4	2.5	6.3	**0.2**
8	5.5	1.2	**0.4**

(a)

	RMS Error Translation [m]		
Seq. #	No Priors	CNN	LiDAR
0	133.9	50.6	**11.9**
2	427.5	92.9	**76.2**
4	32.8	99.9	**3.0**
8	273.4	106.2	**28.0**

(b)

Table 3.2: RMS error in (a) speed and (b) translation for adjustments without and with depth priors for selected sequences. The use of priors greatly decreases errors in both. Results using LiDAR are included for comparison. Note that errors for the method without priors differ from those obtained without objects in Table 2.1 due to a difference in base SLAM system. Scale drift tended to decrease speed to zero, while the system used in Chapter 2 tended to increase speeds, leading to much larger distances between estimated and ground-truth camera positions.

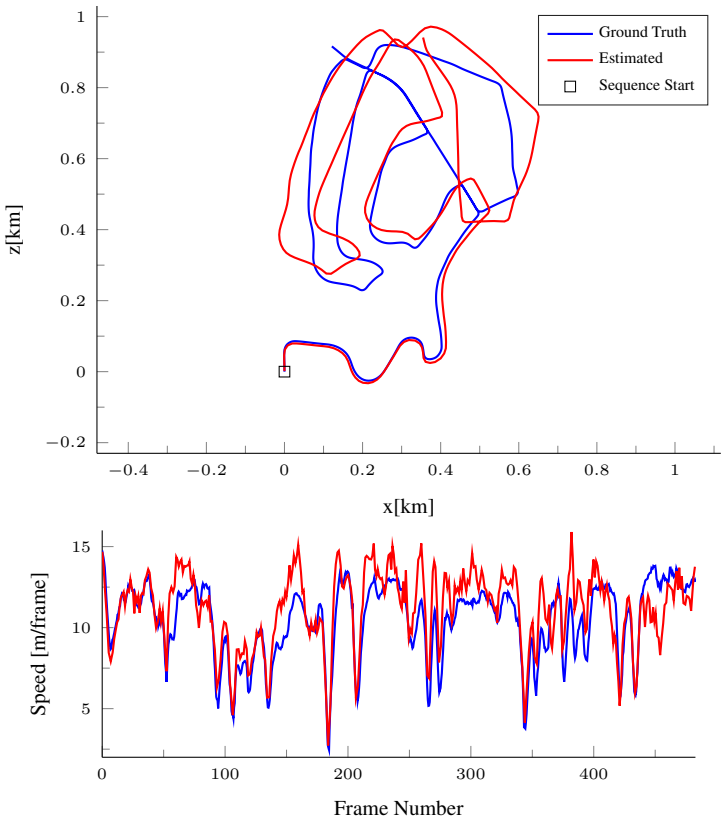


Figure 3.10: Estimated trajectory (top) and speed (bottom) when using the system with image-only depth priors on KITTI sequence 2. While not as accurate as ground-truth depth data, accuracy is significantly greater than the system without priors.

estimated trajectories were both metrically accurate and without scale drift. Next a convolutional neural network that infers depth from a single image was used to provide depths to landmarks. The estimated depths are able to correct scale drift throughout the trajectory, however metric accuracy is not ensured due to the CNNs difficulty with estimating *absolute* depth. While the reasons for this have been discussed, here we discuss a potential avenue of research specific to this domain that might increase accuracy.

As sparse features are used as measurements, estimating the depth for an entire image is unnecessary for this application. A method that estimates depth for all pixels requires learning methods that are able to generalise well, which in turn can lead to less accurate individual depth estimates. By doing without this generality and concentrating training on areas of the image with a priori interesting structure, more accurate depth estimates may be obtained.

Selection of *which* patches to learn could employ the following strategy: First a low-level representation for patches around corners (with associated ground-truth depth) in the image would

be learned. A popular approach to this is the autoencoder [87]. Patches would then be clustered by their representation and only clusters with a low variance in ground-truth depth used for learning. A toy example of this is shown in Fig. 3.11.

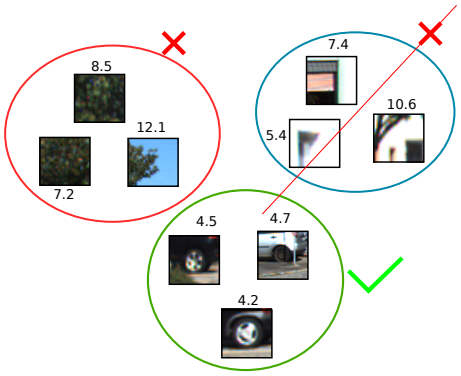


Figure 3.11: A potential solution to selection of patches for depth inference. Patches are clustered according to their low-level representation. Only clusters with a low variance in ground-truth depth are used.

An alternative to this is to directly learn absolute speed of the camera from images, rather than depth. This is investigated in the next chapter.

Chapter 4

A deep speedometer for scale drift correction

This chapter introduces a novel deep convolutional neural network architecture that infers the speed of a camera from monocular imagery. The network is used to place priors on the relative poses predicted from a monocular visual odometry algorithm. The method is tested on a long range dataset and obtains results that are scale drift-free.

4.1 Introduction

In Chapter 2, it was shown that scale drift in monocular SLAM could be reduced by detecting and incorporating objects of a known size into the processing. This imposed a restrictive requirement that multiple objects of the known class were visible frequently throughout operation. A method with far more relaxed requirements was introduced in Chapter 3, where depths at landmarks were predicted using a convolutional neural network (CNN) and used as priors in a modified bundle adjustment. While both of these methods are indeed effective, they attempt to correct scale drift after it has happened by penalising estimated quantities that are affected by scale drift: estimated objects would appear larger or smaller than expected and landmarks would be farther or closer to the camera than expected.

In this chapter, a method for *directly* estimating speed/scale is introduced. We train a deep convolutional neural network to infer the scale of the translation between a pair of images; a term we shall refer to as the per-frame speed of the camera. This information is later used in a visual odometry method as a prior to correct scale drift. As speed is inferred directly it does not require

an accompanying optimisation that penalises estimates of other values to correct scale drift. To demonstrate this, we show how inferred values may easily be incorporated into a simple visual odometry system. This, however, does not restrict it from being used in monocular SLAM. A method for incorporating the inferred speeds as priors in a bundle adjustment is also presented.

Section 4.2 presents a review of work using convolutional neural networks, focusing on works related to the proposed method, such as those used for visual odometry estimation. Section 4.3 details our proposed method for using a CNN for regressing scale from pairs of images. This includes an overview of how a dataset is created for training and methods to augment the dataset to increase its size. More importantly, it details the architecture of the network used for regressing speed estimates. Section 4.4 quantitative results for the method. First a technique for utilising the inferred scale in a simple visual odometry system is presented. To provide context for later results, the odometry method is tested while using ground-truth speed estimates. The network's ability to estimate both absolute and relative speeds is evaluated. A method for reducing noise in estimated speeds is proposed and finally used with the proposed odometry method to estimate trajectories for the given test sets. Methodology for incorporating the inferred scale in a bundle adjustment for monocular SLAM as opposed to monocular odometry is presented in Section 4.5. Finally concluding remarks and future work are discussed in Section 4.6.

4.2 Related work

In this section, a brief history of learning using CNNs is presented. Although originally used for image classification, they have more recently been used for a range of other visual tasks. Some of these are discussed here in context to the proposed method.

In the past, image classification tasks have used hand-crafted features that were extracted from images. These included bags of visual "words" [88] utilising SIFT features [56] or histograms of oriented gradients (HOG) [63]. LeCun *et al.* first showed that CNNs could be trained and used for classification in [29], however it was not until they were applied with great success on the Imagenet challenge by Krizhevsky *et al.* [30] that they became particularly popular. Krizhevsky introduced two main changes to LeCun's networks that proved successful. Firstly they used a new non-linearity known as the rectified linear function (ReL) [89], which did not suffer from vanishing gradients like other non-linear functions and induced sparsity in fully connected layers

of the network. Secondly they used "dropout" [90, 91], a method to prevent over-fitting where random parts of the network are ignored by later stages during training.

More recently, CNNs have been used for more general tasks other than classification and segmentation. Our methodology takes inspiration from the work of Zagoruyko and Komodakis [92] who explore networks architectures for the estimation of a 2D similarity transformation between two images. Three different architectures are explored and compared. These included a Siamese network, where each image is processed by separate convolutional operations the results of which are combined in a fully connected layer at the end of the network; a pseudo-Siamese network, a Siamese network whose convolutional layers share the the same weights; and a two-channel network, where the images are combined into a two-channel input and processed by a single CNN. To deal with multiple image resolutions they utilise a spatial pyramid pooling layer (SPP), which was originally implemented in [93]. They show that the features extracted from a two-channel input provide better information than features obtained from separate features obtained from Siamese networks. Because of this, we choose to use the same network structure in our method; combining two RGB images into a single 6-channel input image.

Fischer *et al.* [94] apply deep learning methods to the estimation of optical flow between two images. They compare two different network architectures, a simple convolutional network using a two-channel input, and a novel Siamese-style network that attempts to determine the correlation between features for each "wing" of the network. These are used explicitly for matching operations used in the computation of optical flow. Slightly better results are obtained using the novel-network. Despite this success, there is no guarantee that this matching is involved in the estimation of speed and therefore we keep the proposed two-channel architecture.

A similar work is that of Konda and Memisevic [95] who apply deep learning methods for visual odometry. The proposed network processes the five most recent frames from both the left and right stereo images of the KITTI dataset. These are each combined in a Siamese-style network that predicts *discretised* camera motion, effectively reducing the problem to a classification task. The authors note that the amount of data in the KITTI dataset proved insufficient for full regression.

4.3 Scale regression from image pairs

In this section we detail how a CNN is trained to estimate the speed of the camera from pairs of images. First the data used for inference is introduced in Section 4.3.1. This includes inputs to the network, and the target values for each input. Some strategies for increasing the size of the training set are also detailed. Section 4.3.2 describes the actual CNN used for inference, while the training regime is described in Section 4.3.3. Finally, conjecture about *how* the network might infer speed is discussed in Section 4.3.4.

4.3.1 Dataset generation

The selected dataset used for training is the KITTI outdoor odometry dataset [72, 50]. The first 11 sequences from this dataset come with ground truth poses, and will therefore be used for training and testing. Three difficult sequences; namely 00, 02, and 08, from the dataset were selected for testing, while the remaining 8 were used for training the network. These sequences contain about 10000 training examples in total.

Input images

The inputs to the proposed CNN is a stacked pair of images each 240×120 pixels in size. The KITTI dataset contains frames that vary in size, but most are about 1241×376 pixels. While simply re-sizing all images to the input size might seem appropriate, it ignores the fact that different sequences have different intrinsic camera calibrations. For this reason, we normalise all images to a single calibration matrix $\mathtt{K}_{\text{norm}}$. For each homogeneous pixel coordinate $\mathbf{u}$ in the new normalised image I_{norm}, the corresponding colour in the original image is computed and used:

$$I_{\text{norm}}(\mathbf{u}) \leftarrow I_{\text{orig}}(\mathtt{K}_{\text{orig}}\mathtt{K}_{\text{norm}}^{-1}\mathbf{u}) \tag{4.1}$$

where $\mathtt{K}_{\text{orig}}$ and I_{orig} are the original calibration matrix and image respectively. Sub-pixel interpolation is considered unnecessary, and so $\left(\mathtt{K}_{\text{orig}}\mathtt{K}_{\text{norm}}^{-1}\mathbf{u}\right)$ is simply rounded to the nearest integer.

(a) (b)

Figure 4.1: (a) Original image before normalisation and (b) image after normalisation which is used as input to the proposed CNN.

The target intrinsic calibration matrix $\mathtt{K}_{\text{norm}}$ is:

$$\mathtt{K}_{\text{norm}} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 300 & 0 & 120 \\ 0 & 300 & 60 \\ 0 & 0 & 1 \end{bmatrix} . \tag{4.2}$$

The principal point (u_0, v_0) was chosen as the centre of the output image, and the focal lengths (f_x, f_y) were chosen to ensure that most of the original image was used and that all normalised pixels have a corresponding pixel in the original image. Fig. 4.1 shows the result of this normalisation. The process does result in some loss of information at the edges of the image, however, most of the information remains intact.

Training labels

Given a pair of input images $I_i()$ and $I_j()$ with ground-truth poses $\mathtt{T}_i$ and $\mathtt{T}_j$, the relative pose between them is computed as:

$$\mathtt{T}_{ij} = \mathtt{T}_j \mathtt{T}_i^{-1} \tag{4.3}$$

The target output for the image pair is the speed of the camera, measured as

$$s_{ij} = \|\mathbf{t}_{ij}\| \ , \tag{4.4}$$

where $\mathbf{t}_{ij}$ is the translational component of $\mathtt{T}_{ij}$. Note that speed is measured on a *per-frame* rather than a *per-second* basis. This is to account for potential differences in frame-rates of different datasets.

Dataset selection and augmentation

The sequences used for training contain about 10000 training examples in total. The following strategies are employed to increase the size of the training set:

- The KITTI dataset comes with stereo images for stereo odometry and SLAM. Images from both cameras are utilised.
- Each training pair of images has their order reversed (as if the camera were moving backwards), while keeping their target value the same.
- Each image from a training pair is used to create a new training pair with the same image twice as input to simulate a stationary camera. The target value for these new pairs is 0.

This increases the size of the dataset to about 80000 examples. Additionally, dropout is applied to convolutional layers and fully connected layers with a 30% chance during training to prevent over-fitting.

4.3.2 Network overview

Fig. 4.2 shows the network architecture used. The network takes as its input a 6-channel image consisting of two images stacked together. This is fed into three 3×3 convolutional layers where each convolution is followed by a 2×2 max-pooling layer. These are followed by three fully connected layers. The rectified linear function (ReL) is used as a non-linearity between layers:

$$f(x) = \max(x, 0) \,, \tag{4.5}$$

and the training loss used is a Euclidean loss:

$$E(s, \hat{s}) = (s - \hat{s})^2 \,, \tag{4.6}$$

where $\hat{s}$ is the predicted output of the network.

4.3.3 Training procedure

The network was implemented using the Theano framework [96], and trained using back-propagation using gradient-descent. Training took place over 50 epochs where the step size started at 0.01 and

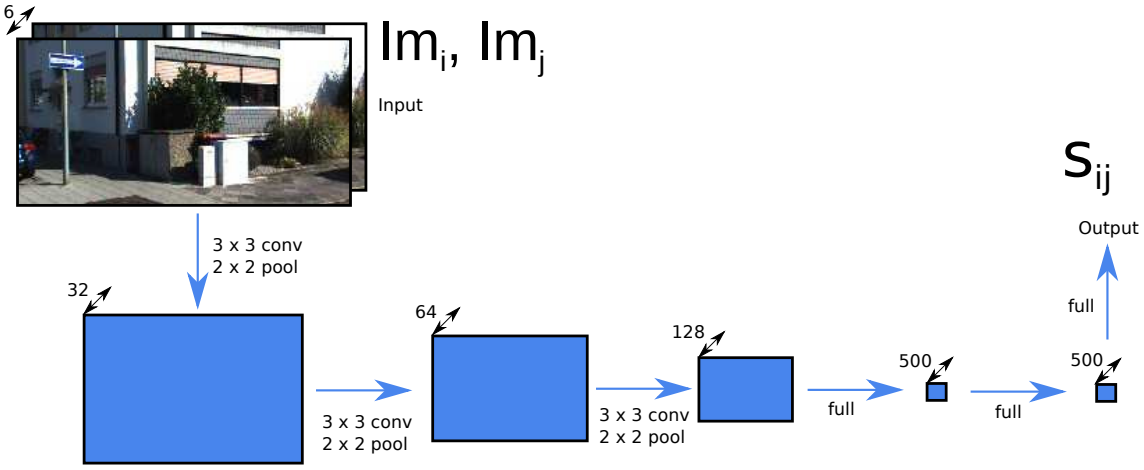


Figure 4.2: Network architecture used for speed regression.

was gradually reduced, eventually ending at 0.00001 by the last epoch. Training took approximately 8 hours to converge using a Nvidia GTX TITAN X. In terms of feed-forward performance, the network can process a sequence of 4540 pre-processed examples in approximately 3 seconds.

4.3.4 Inner-workings of the CNN

The exact details of *how* CNNs perform inference is still not fully understood [97, 98]. Despite this, a discussion of how the network may infer speed is warranted. We postulate that the network could be learning to extract a rudimentary form of optical flow between the images, as well as an absolute depth estimate. The camera speed is then inferred by internally accumulating this information.

4.4 Evaluation

The method for a data-driven approach for speed estimation using convolutional neural networks has been proposed. This section includes evaluation of the method on a long range dataset. Section 4.4.1 details how the proposed method is incorporated easily into an existing visual odometry method. In Section 4.4.2 the method is first tested using ground truth speeds to provide context for later results. Section 4.4.3 provides quantitative evaluation of the method, and provides insights into areas where estimation is potentially inaccurate. Section 4.4.4 introduces the use of a Kalman filter on top of the network output, allowing for more correlated speed estimates, and provides estimated monocular trajectories on the given test sequences.

4.4.1 System overview

To test the estimated speed, we utilise an existing visual odometry method [49] that computes relative poses between frames. Given a relative estimate between the previous and current frame $\mathtt{T}_{\text{relative}}$, the current tracker pose is updated every frame using:

$$\mathtt{T}_C \leftarrow \mathtt{T}_{\text{relative}}\mathtt{T}_C \ . \tag{4.7}$$

Like many other monocular odometry methods, the original visual odometry algorithm estimates the correct scale of the relative pose estimate by using a fixed-camera height assumption. To remove this assumption, the translation component $\mathbf{t}_{\text{relative}}$ of the relative pose estimate is re-scaled as follows:

$$\mathbf{t}_{\text{relative}} \leftarrow \mathbf{t}_{\text{relative}} \frac{s}{\|\mathbf{t}_{\text{relative}}\|} \ , \tag{4.8}$$

where s is the estimated speed from the CNN. This is obviously done *before* updating the pose as shown in Eq. (4.7).

4.4.2 Monocular visual odometry with ground-truth speed

Even if the given network predicts scale perfectly, the odometry method used is subject to rotational and translational drift. We provide an example of an estimated trajectory with *perfect* scale to show the maximum accuracy of the trajectory that is obtainable. Fig. 4.3 therefore shows the estimated trajectory for KITTI sequence 00 with ground-truth speed estimates.

4.4.3 Quantitative results

The estimated speed using the suggested method is now evaluated. For quantitative evaluation we use two error metrics. Given a set of n predictions, the absolute error in speed is defined as:

$$E_{\text{abs}} = \sum_{i=0}^{n} \sqrt{(s_i - \hat{s}_i)^2} \ , \tag{4.9}$$

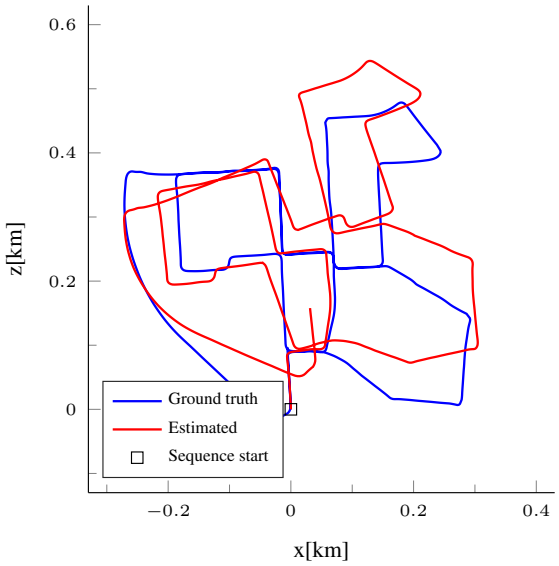


Figure 4.3: Estimated trajectory for KITTI sequence 00 using proposed odometry method augmented with ground-truth speed estimates.

Sequence	E_{abs}[m/frame]	$E_{\text{scale-inv}}$[m/frame]	Scale factor (α)
00	0.43	0.24	1.78
02	0.62	0.25	2.08
08	0.38	0.19	1.65

Table 4.1: Absolute error, scale-invariant error, both in m per frame, and scale-factor for test sequences.

where s_i and $\hat{s}_i$ are the ground-truth and estimated speeds of the ith example respectively. A scale-invariant speed measures the ability of the network to estimated *relative* speeds

$$E_{\text{scale-inv}} = \sum_{i=0}^{n} \sqrt{(s_i - \alpha \hat{s}_i)^2}\,, \tag{4.10}$$

where α is a scaling parameter chosen such that $E_{\text{scale-inv}}$ is minimised. Table 4.1 shows the absolute error and scale-invariant error in speed for the three test sequences as well as the scaling parameter used for scale-invariant error. The low scale-invariant error shows that the network is very good at estimating *relative* speeds. Absolute error, however, is slightly higher. This result does seem to mirror results from Chapter 3, where deep networks excelled at relative rather than absolute depth estimation. While the network is not able to resolve absolute scale exactly, its ability to estimate relative speeds accurately is enough to warrant its use for scale drift reduction. We postulate that more accurate scale estimation is possible provided more training data.

Fig. 4.4 shows the estimated per-keyframe speed versus ground truth, scaled by their respective

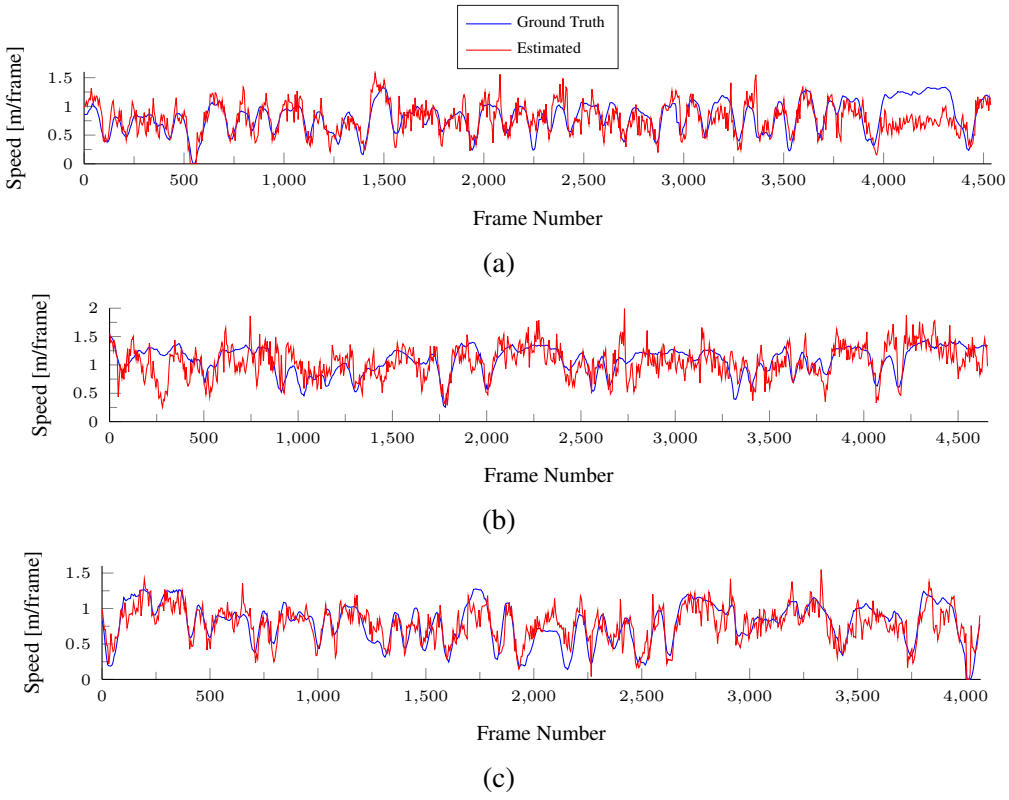


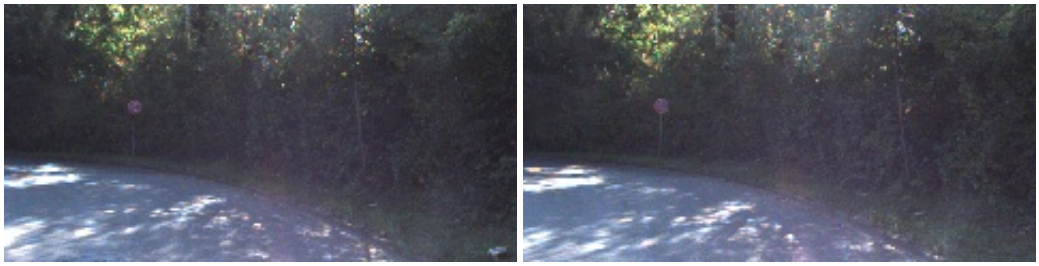
Figure 4.4: Estimated per-keyframe speed vs ground-truth for KITTI (a) sequence 00, (b) sequence 02, and (c) sequence 08 using the proposed CNN.

scale factors in Table 4.1. Despite some noisy measurements, the speed estimate does seem to be able to estimate the ground truth speed correctly up to scale. Noteworthy examples are at around frames 500 in Fig. 4.4(a), where the network is able to correctly determine the camera coming down to a stop and speeding up again. The network understandably battles with very similar image pairs, believing them to be the result of a slow or stationary camera. This can be seen starting at frame 4000 in Fig. 4.4(a) and frame 250 in Fig. 4.4.(b). The specific images that resulted in these estimates are shown in Fig. 4.5.

4.4.4 Noise reduction using a Kalman filter

As each pair of images are processed independently, past speed estimates are effectively ignored for each estimation. While there might be inherent benefits in this (a method that is dependent on past estimates might itself drift), the speed of the camera at a particular frame is highly correlated with the speeds of past and future frames in reality. For this reason, we use a Kalman filter on top of speed estimates, which assumes that the speeds estimated from the network are noisy

(a)



(b)

Figure 4.5: Image pairs that result in a large error in estimated speed (a) from sequence 00 at frame 4000 and (b) from sequence 02 at frame 250. Both pairs include very similar images resulting in a lower speed estimate.

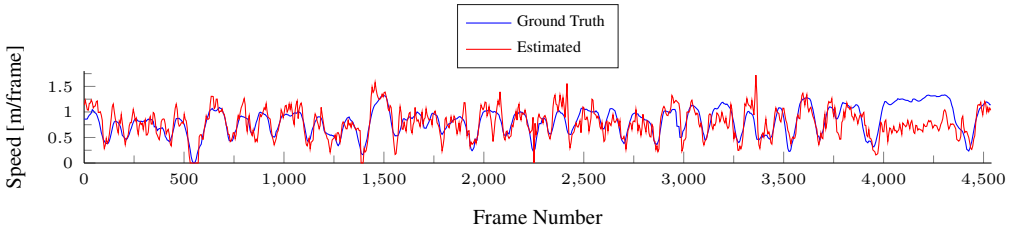


Figure 4.6: Estimated per-keyframe speed vs ground-truth for sequence 00 while using a Kalman filter.

measurements of some underlying true speed.

For the given test sequences, the benefit of a Kalman filter seems to be purely cosmetic, resulting in a slight smoother speed estimate as shown in Fig. 4.6, while not significantly reducing estimated speed error. That being said, the scenario where its use allows for larger sporadic errors to be ignored is entirely possible.

Finally, the filtered estimated speeds are used with the proposed odometry method on the three test sequences. Fig. 4.7 shows the resultant estimated trajectories compared to ground truth. Despite some translational and rotational drift, scale drift is corrected for most of the sequences. Note that speeds used are scaled up by their relevant scale factors in Table 4.1.

4.5 A remark on bundle adjustment

In this chapter, speed estimates from the network have been incorporated into visual odometry by directly setting the size of the relative translation estimates each frame. The speed estimates are, however, not constrained to visual odometry, and may be applied to visual SLAM. This section details how this may be achieved.

Estimated speeds are incorporated into a bundle adjustment by placing pose-pose constraints between keyframes. The set of landmarks $\mathcal{X}$ and keyframes $\mathcal{T}$ that minimises the following error is found

$$\{\mathcal{X}, \mathcal{T}\} = \underset{\{\mathcal{X},\mathcal{T}\}}{\arg\min} \left[E_{\text{reproj}}(\mathcal{X}, \mathcal{T}) + E_{\text{pose}-\text{pose}}(\mathcal{T}) \right] \ , \tag{4.11}$$

where E_{reproj} is the reprojection error between keyframes and landmarks. The pose to pose error $E_{\text{pose}-\text{pose}}$ places relative constraints between keyframes, and penalises frames that do not adhere to this constraint. For two keyframes $\mathtt{T}_k$ and $\mathtt{T}_l$ with a relative pose constraint $\mathtt{T}_{k,l}$ the pose-graph residual [75, 55] is:

$$\mathbf{r}_{kl} = \log_{SE(3)}(\mathtt{T}_{k,l}\mathtt{T}_k\mathtt{T}_l^{-1}) \ , \tag{4.12}$$

where $\log_{SE(3)}()$ is the *inverse* of the exponential map, being a map from $\mathbf{SE}(3)$ to $\mathfrak{se}(3)$. When the pose constraint is adhered to $\mathtt{T}_k\mathtt{T}_l^{-1} = \mathtt{T}_{k,l}^{-1}$, the error is then:

$$\mathbf{r}_{kl} = \log_{SE(3)}(\mathtt{I}) = \mathbf{0} \tag{4.13}$$

To introduce estimated speeds from the CNN, pose constraints can initially be computed from the original keyframe poses:

$$\mathtt{T}_{k,l} = \mathtt{T}_l^{-1}\mathtt{T}_k \ , \tag{4.14}$$

The translational components of these constraints $\mathbf{t}_{k,l}$ are then re-scaled by the estimated speeds s using Eq. (4.8). The network might find it difficult to estimate speeds for larger displacements from keyframes that are further apart. In this situation, the pose constraint may be obtained by accumulating a kinematic chain of the frames between them:

$$\mathtt{T}_{k,l} = \Pi_{i=0}^{n}\mathtt{T}_i \tag{4.15}$$

where $\mathtt{T}_i \in \mathtt{T}_0, \mathtt{T}_1, ...\mathtt{T}_n$ are *relative* pose estimates obtained from the tracking system between keyframes $\mathtt{T}_k$ and $\mathtt{T}_l$ each re-scaled using a speed estimate obtained from the network.

4.6 Conclusion

In this chapter, a data-driven approach to estimating speed between pairs of frames has been introduced. The presented convolutional neural network was trained on a subset of the KITTI odometry dataset, and tested on three difficult sequences. Although the estimated trajectories were not metrically accurate, the network was able to estimate relative speed correctly, and therefore correct scale drift.

As the method is designed for temporal data, an interesting avenue of research would be to apply recurrent networks to speed estimation. In this way, past estimates of speed would be able to aid current estimates effectively.

While the presented network is an effective proof-of-concept, the amount of training data and the network size is comparatively small compared to modern deep networks. This shows its inability to accurately estimate absolute speeds accurately. Additionally the network has been training on a specific domain. For the KITTI dataset, the camera is always moving in a forward direction, and therefore it is unlikely that the network in its current state will be able to adapt to situations such as a laterally moving camera. Future research could include training on a much wider set of domains and testing larger network structures. A benefit of this avenue of research is that training data is effectively "free". By using a camera with a properly paired IMU, the size of the training dataset that can be obtained is effectively limitless.

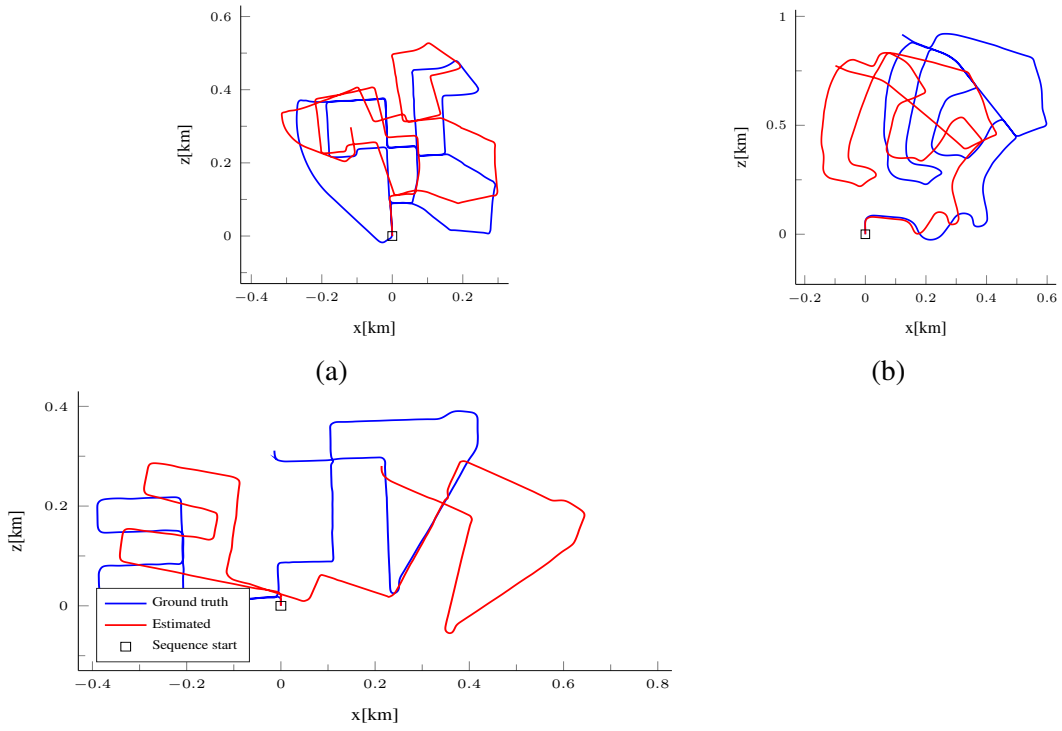


Figure 4.7: Estimated trajectories for KITTI (a) sequence 00, (b) sequence 02, and (c) sequence 08 using a Kalman filter whose measurements are speed estimates from the proposed CNN.

Chapter 5

A sparsified direct camera tracker

This chapter introduces a sparsified direct camera tracker that operates through alignment of image patches. A novel affine warp, unique to patches, is introduced that allows computational costs to be reduced. This in turn allows for multiple reference keyframes to be used in tracking, increasing accuracy. The method is compared to a state of the art tracker that uses high image-gradient regions, and obtains equal if not more accurate results.

5.1 Introduction

Real-time tracking has often relied on *sparse* feature extraction and description [4, 5, 9, 31, 18] to limit information dealt with to ensure real-time operation. Rather than operating on extracted features, *direct methods* estimate map-structure and camera motion directly from image intensities [12, 14, 32, 33, 35]. While more computationally expensive, they offer a number of advantages over feature-based methods, such as invariance to motion blur and defocus [14] and robustness in areas with little texture [99].

Although direct depth estimation from image intensities is a well-studied problem [10], it was not until GPUs became available that methods could be routinely parallelised and deliver real-time performance [11, 12]. An interesting new development is the number of works that have sought to remove the requirement of a GPU by limiting depth-estimation to areas of high gradient [35], or sparse corners [34] in the image. Both of these works estimate the current pose of the camera by maintaining an estimate of a relative transformation between the current frame and either the previous frame [34] or a single reference frame [35].

The purpose of this chapter is to introduce a direct camera tracking method that has a low

computational cost by utilising sparse corners in an image. In doing so, it achieves speeds that are twice as fast as a method that uses high-gradient regions in the image [16]. In this way it is closest to the semi-direct visual odometry (SVO) method introduced in [34], but some key differences should be noted. The method is a tracker for a SLAM method rather than a visual odometry tracker. Rather than estimate relative poses between consecutive poses, as is the case with SVO [34], the proposed method estimates a *global* camera pose by tracking relative to keyframes that are part of a map. As SVO estimates small relative changes to camera pose, the direct implementation does not take into account changes to patch appearance due to perspective distortion as this is negligible from frame to frame. In the proposed method, perspective distortion is likely to occur between the current frame and a reference keyframe due to a larger difference in global pose, and a warp is estimated for each patch to accommodate for the distortion. Additionally, the method presented uses an naïve inverse additive formulation that allows template pixels to be pre-interpolated. The intention of this formulation is a slight increase in performance.

While [34, 16] estimate the current camera frame relative to a *single* reference keyframe, the proposed method's lower computational cost allows us to use *multiple* reference keyframes while still maintaining real-time performance. The additional multiple-viewpoint information leads to a more globally consistent and accurate camera estimate, especially in more challenging conditions such as video with strong image blur.

Section 5.2 presents an overview of related methods that operate through direct image alignment. Section 5.3 details the proposed sparse direct tracking method. Details of pose refinement using direct image alignment are discussed, as well the specific details of how they are used in the proposed tracker. Section 5.4 presents strategies that are used to ensure accurate camera tracking. Experiments comparing the proposed tracker to a current state of the art method are presented in Section 5.5. Concluding remarks are presented in Section 5.6.

5.2 Related work

Direct estimation of depth from images is a mature and well-studied problem. All direct methods are founded on the same underlying assumption of brightness constancy: the same part of the scene when viewed in different images has a constant irradiance. Okutomi and Kanade [10] originally estimated dense-depth in a stereo matching problem for a set of multiple baseline cameras using a

sum of squared differences (SSD) of observed and expected pixel intensities for small patches in the images. They proved that when using the sum of SSDs (SSSD) over a number of frames any ambiguity in a point's depth could be resolved, even in the case of repeating patterns in the image. Sato *et al.* [100] used this approach for a structure from motion problem, modifying the SSSD measure to utilise the the median of SSDs for a particular point, giving increased outlier rejection.

Initial works [11, 12] on real-time direct reconstruction from monocular images constructed a 3D surface from a set of images taken from known viewpoints, although both still required sparse 3D points to track the camera around the scene. Newcombe and Davison [11] created an initial base mesh from interpolated sparse 3D points, then updated the mesh from optical flow measurements. Stuhmer *et al.* [12] proposed a variational method which estimates a depth-map that preserves image intensities in multiple images. The depth-map was smoothed using a total variation regulariser that removes noise while still preserving edges. In DTAM, Newcombe *et al.* built on this by utilising the resultant dense 3D model not only for mapping but dense tracking as well [14]. Pizzoli *et al.* [33] also used a variational implementation, however the optimisation proposed serves only as a spatial regulariser on depth estimates from probabilistic per-pixel filters [32]. All of these real-time methods attempt to estimate depth for the majority of pixels in a given image, a highly parallelisable task suitable to computation on a GPU.

While the parallel processing capacity of a GPU is able to handle these computations and maintain real-time performance, there has been recent interest in developing direct methods that estimate depth for the most salient parts of the image, thereby reducing computational load and allowing for real-time operation on a CPU. One such work is that of Engel *et al.* [35]. They use direct image alignment and per-pixel inverse-depth filters and to estimate depth in select areas in the image with high-gradient. In Large-Scale Direct Monocular(LSD)-SLAM [16], the method is extended with a scale-aware pose graph optimisation [75], which makes use of scale estimation that takes place during image alignment between keyframes.

Another is SVO [34] introduced by Forster *et al.* but they constrain direct depth-estimation to even sparser corners in the image. Like [35, 16], depth is initially estimated using depth filters. The description semi direct refers to the following. Camera tracking first refines the camera pose using a direct image-alignment that minimises photometric error. This is followed by an additional refinement that minimises geometric error.

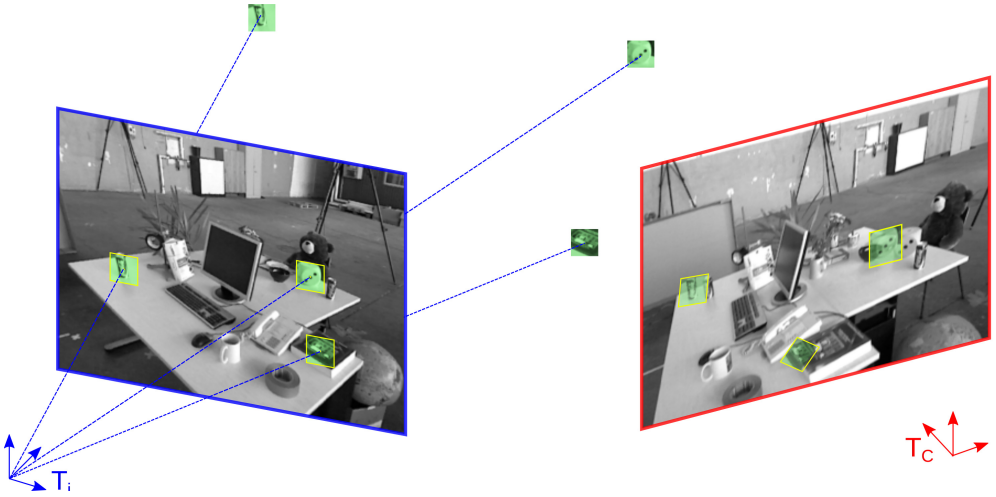


Figure 5.1: Pose refinement using direct image alignment. The keyframe shown in blue has a number of patches (shown in green) with associated depth, allowing them to be un-projected into 3D points in the map. These may be projected into the current camera (shown in red). The warped patches are shown in green in the image. The current camera's pose $\mathtt{T}_C$ is refined to align the projected patches (green) with the current image. In the example case, the current camera should move to the left of its current centre to minimise the difference between patch and image values.

5.3 Pose refinement using direct image alignment

Tracking using direct image alignment refines the current camera estimate to minimise a *photometric* error term, which penalises errors in pixel intensities, rather than a *geometric* error term, which penalises the difference between a reprojection of a landmark and its measurement.

An example is shown in Fig 5.1, where three corners with associated depth are present in the keyframe with pose $\mathtt{T}_i$ shown in blue. Each corner is associated with a patch shown in green in the keyframes image and gives rise to a point in 3D space due to its depth. When refining the current camera pose $\mathtt{T}_C$, shown in red, the corners' associated points in 3D space are re-projected into the current image, and the patches warped appropriately to deal with perspective distortion. The *photometric* error is the summed squared difference between intensities in the pixels under which the patches lie, and those in the pixels themselves. As the pose $\mathtt{T}_C$ is refined, this changes the reprojection and the warping of the patches, minimising this difference.

This section begins by introducing direct image alignment for *independent* patches in the image in Section 5.3.1, whose position in an image are not a result of a reprojection. Next, patches are considered the result of a reprojection of a 3D point and a method for refining pose using direct image alignment with any given warp is detailed in Section 5.3.2. One such warp is a projective warp, which is discussed in Section 5.3.3. An approximation to this is the affine warp,

which is discussed in Section 5.3.4. A further approximation is the inverse additive formulation, which allows for some gradient terms to be pre-computed before optimisation. This is discussed in Section 5.3.5.

5.3.1 Direct image alignment

In this section *independent* image alignment is introduced. Here, the position of a patch is not the result of a reprojection, but is able to move freely in the image. The original Lucas-Kanade [101] algorithm is used to align a template image $I_{\text{template}}(\mathbf{u})$ to part of an input image $I_{\text{target}}(\mathbf{u})$. It does so by finding an offset $\mathbf{p} \in \mathbb{R}^2$ to the target image that minimises the sum of pixel differences (SSD) between the template and a portion of the image

$$E_{SSD} = \sum_{\mathbf{u} \in \Omega} \left[I_{\text{target}}(\mathbf{u} - \mathbf{p}) - I_{\text{template}}(\mathbf{u}) \right]^2 \ , \tag{5.1}$$

where Ω is the set of pixels containing the region of interest to be tracked. The translation $\mathbf{p}$ can be generalised by a warp $W(\mathbf{u}, \mathbf{p})$ parameterised by $\mathbf{p}$, which allows for changes in perspective to be adequately modelled:

$$E_{SSD} = \sum_{\mathbf{u} \in \Omega} \left[I_{\text{target}}(W(\mathbf{u}, \mathbf{p})) - I_{\text{template}}(\mathbf{u}) \right]^2 \ . \tag{5.2}$$

5.3.2 Pose refinement: general form

The problem of refining camera pose from direct image alignment is now addressed. Here, patches are no longer able to move freely, but are dependent on an estimated camera pose. A sparse map is assumed, consisting of a set of keyframes $\mathcal{K}$ each associated with a set of corners $\mathbf{v}_j \in \mathcal{C}_i$ for which a depth d_j is known. The assignment of depth to corners is discussed later in Section 5.5. Each keyframe i is associated with a pose $\mathtt{T}_i \in \mathbf{SE}(3)$ and an image $I_i()$. The template associated with each corner is extracted from the set of pixels Ω_j around its centre $\mathbf{v}_j$. This is commonly, but not restricted to, a set of pixels from a square patch centred at $\mathbf{v}_j$.

For real time tracking, the pose of the current frame $\mathtt{T}_C$ with image $I_C()$ is to be estimated. An initial pose is given from the frame's previous position and a motion model. The goal is then

to find a pose that minimises

$$E = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_C(W(\mathbf{u}_k, d_j, \mathtt{T}_C, \mathtt{T}_i)) - I_i(\mathbf{u}_k) \right]^2 = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} r_{ijk}^2 \, . \tag{5.3}$$

This is equivalent to finding a maximum likelihood estimate of $\mathtt{T}_C$ provided photometric differences r_{ijk} are i.i.d. and normally distributed. For mathematical simplicity, the case of a single corner $\mathbf{v}_j$ from a single keyframe i is considered first. Later this is be generalised to multiple keyframes each with multiple corners. For a single corner and keyframe Eq. (5.3) becomes:

$$E = \sum_{\mathbf{u}_k \in \Omega_j} \left[I_C(W(\mathbf{u}_k, d_j, \mathtt{T}_C, \mathtt{T}_i)) - I_i(\mathbf{u}_k) \right]^2 \, . \tag{5.4}$$

Eq. (5.4) is minimised by successively applying perturbations to the pose $\mathtt{T}_C$. Perturbations are parameterised by vector $\boldsymbol{\mu} \in \mathfrak{se}(3)$ and applied as follows:

$$\mathtt{T}_C \leftarrow \exp(\boldsymbol{\mu}) \mathtt{T}_C \, . \tag{5.5}$$

The goal is then to find

$$\boldsymbol{\mu} = \arg\min_{\boldsymbol{\mu}} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_C(W(\mathbf{u}_k, d_j, \exp(\boldsymbol{\mu}) \mathtt{T}_C, \mathtt{T}_i)) - I_i(\mathbf{u}_k) \right]^2 \, . \tag{5.6}$$

Linearising around $\mathtt{T}_C$

$$E = \sum_{\mathbf{u}_k \in \Omega_j} \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) + \nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \boldsymbol{\mu} - I_i(\mathbf{u}_k) \right]^2 \, , \tag{5.7}$$

where ∇I_C is the gradient of the current image. Differentiating with respect to $\boldsymbol{\mu}$ and setting to zero

$$\frac{\partial E}{\partial \boldsymbol{\mu}} = \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \right]^\top \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) + \nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \boldsymbol{\mu} - I_i(\mathbf{u}_k) \right] = 0 \, . \tag{5.8}$$

Note that ∇I_C is evaluated at $W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)$. Solving for $\boldsymbol{\mu}$:

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \right]^\top \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) - I_i(\mathbf{u}_k) \right] \, , \tag{5.9}$$

where $\mathtt{H}$ is

$$\mathtt{H} = \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}}\right]^\top \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}}\right] . \tag{5.10}$$

For multiple corners and keyframes, $\boldsymbol{\mu}$ and $\mathtt{H}$ are the sums of the contributions from each individual corner.

$$\begin{aligned} \boldsymbol{\mu} &= \mathtt{H}^{-1} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}}\right]^\top [I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) - I_i(\mathbf{u}_k)] \\ &= \mathtt{H}^{-1} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}}\right]^\top r_{ijk} , \end{aligned} \tag{5.11}$$

where $\mathtt{H}$ is

$$\mathtt{H} = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}}\right]^\top \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}}\right] . \tag{5.12}$$

5.3.3 Projective warp

The method for pose refinement using a *general* warp from one frame to another has been introduced. In this section, details of the *projective* warp are introduced. For a patch with a known depth, the projective warp first produces a point in 3D space for *each* pixel in the patch. The warped patch is the result of these 3D points being re-projected into the new image.

For a pixel $\mathbf{u}_k$ from a patch j with depth d_j in the frame with pose $\mathtt{T}_i$, the resultant 3D point is:

$$\mathbf{X}_{i,j,k} = \mathbf{X}(\mathbf{u}_k, d_j, \mathtt{T}_i) = \mathrm{proj}_K^{-1}(\mathbf{u}_k, d_j, \mathtt{T}_i) , \tag{5.13}$$

where proj_K^{-1} is the combined back-projection function introduced in Eq. 2.8. The projective warp takes $\mathbf{X}_{i,j,k}$ and projects it into the image of the frame with pose $\mathtt{T}_C$:

$$W_{\mathrm{proj}}(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i) = \mathrm{proj}_K(\mathbf{X}_{i,j,k}, \mathtt{T}_C) . \tag{5.14}$$

For a small offset to $\mathtt{T}_C$:

$$W_{\mathrm{proj}}(\mathbf{u}_k, \exp(\boldsymbol{\mu})\mathtt{T}_C, \mathtt{T}_i) = \mathrm{proj}_K(\mathbf{X}_{i,j,k}, \exp(\boldsymbol{\mu})\mathtt{T}_C) . \tag{5.15}$$

Eq. (5.11) and Eq. (5.12) require a derivative of the warp with respect to the pose perturbation $\frac{\partial W}{\partial \boldsymbol{\mu}}$. For the projective warp, this is the derivative the projection of point $\mathbf{X}_{i,k,k}$ in a frame with respect

to a change in that frames pose. This is the 2×6 Jacobian matrix $\mathtt{J}_{\text{projcam}}$:

$$\frac{\partial W}{\partial \boldsymbol{\mu}} = \mathtt{J}_{\text{projcam}}(\mathbf{X}_{i,j,k}, \exp(\boldsymbol{\mu})\mathtt{T}_C) \ . \tag{5.16}$$

The formula for the derivative $\mathtt{J}_{\text{projcam}}$ is well known and derived in Appendix A. Eq. (5.11) and Eq. (5.12) then become:

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \mathtt{J}_{\text{projcam}}\right] r_{ijk} \ , \tag{5.17}$$

where $\mathtt{H}$ is

$$\mathtt{H} = \sum_i \sum_j \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \mathtt{J}_{\text{projcam}}\right]^\top \ . \left[\nabla I_C \mathtt{J}_{\text{projcam}}\right] \tag{5.18}$$

5.3.4 Affine warp

It should be noted here that LSD-SLAM [16] uses a projective warp, as every pixel in a reference image has its own depth. By contrast, here we assume that every pixel in the same patch share the same depth, which allows for use of an affine warp. The affine warp assumes that the projection function is approximately linear around the centre of the patch.

The performance benefits of this are twofold. Firstly the projection from one frame to another need only be applied once to the centre of each patch; and the other pixels can be computed using the affine warp. Secondly, all pixels in the patch share the same camera-pose dependent term; allowing for factorisation of the update equations.

To compute the affine warp for each patch, the inverse projection and then projection of its centre pixel $\mathbf{v}_j$ in its source keyframe is computed. Note that this only computed for a single pixel.

$$\mathbf{v}'_j = W_{\text{proj}}(\mathbf{v}_j, \mathtt{T}_C, \mathtt{T}_i) = \text{proj}_K(\text{proj}_K^{-1}(\mathbf{v}_j, d_j, \mathtt{T}_i), \mathtt{T}_C) \ . \tag{5.19}$$

The projection function is then linearised around $\mathbf{v}_j$ for a small pixel offset $\boldsymbol{\psi}$:

$$W_{\text{proj}}(\mathbf{v}_j + \boldsymbol{\psi}, \mathtt{T}_C, \mathtt{T}_i) \approx W_{\text{proj}}(\mathbf{v}_j, \mathtt{T}_C, \mathtt{T}_i) + \frac{dW_{\text{proj}}}{d\mathbf{v}_j}\boldsymbol{\psi} \tag{5.20}$$

$$\approx \mathbf{v}'_j + \frac{dW_{\text{proj}}}{d\mathbf{v}_j}\boldsymbol{\psi} \ . \tag{5.21}$$

For speed and simplicity, $\frac{dW_{\text{proj}}}{d\mathbf{v}_j}$ is approximated using a first order forward difference, which forms the matrix $\mathtt{A}$:

$$\frac{dW_{\text{proj}}}{d\mathbf{v}_j} = \left[\frac{\partial W_{\text{proj}}}{\partial v_1}, \frac{\partial W_{\text{proj}}}{\partial v_2}\right] = \mathtt{A} \ , \tag{5.22}$$

where

$$\begin{aligned} \frac{\partial W_{\text{proj}}}{\partial v_1} &\approx W_{\text{proj}}(\mathbf{v}_j + [1,0]^\top, \mathtt{T}_C, \mathtt{T}_i) - W_{\text{proj}}(\mathbf{v}_j, \mathtt{T}_C, \mathtt{T}_i) \\ \frac{\partial W_{\text{proj}}}{\partial v_2} &\approx W_{\text{proj}}(\mathbf{v}_j + [0,1]^\top, \mathtt{T}_C, \mathtt{T}_i) - W_{\text{proj}}(\mathbf{v}_j, \mathtt{T}_C, \mathtt{T}_i) \ . \end{aligned} \tag{5.23}$$

A photometric difference for a pixel corresponding to an offset $\boldsymbol{\psi}$ from the centre $\mathbf{v}_j$ in the patch is now computed as follows:

$$I_C(\mathbf{v}' + \mathtt{A}\boldsymbol{\psi}) - I_i(\mathbf{v} + \boldsymbol{\psi}) \ . \tag{5.24}$$

Note that $\boldsymbol{\psi}$ does not lie in the same coordinate system as pixels $\mathbf{u}_k$, which has its origin at the top left corner of the image, but at the coordinate frame centred at $\mathbf{v}$. A pixel $\mathbf{u}$ with coordinates in the original coordinate frame is:

$$\mathbf{u} = \mathbf{v} + \boldsymbol{\psi} \ . \tag{5.25}$$

Replacing $\boldsymbol{\psi}$ in Eq. (5.24), the photometric difference is

$$I_C(\mathbf{v}' + \mathtt{A}(\mathbf{u}_k - \mathbf{v})) - I_i(\mathbf{u}_k) \ , \tag{5.26}$$

and rearranging it becomes:

$$I_C(\mathtt{A}\mathbf{u}_k - \mathtt{A}\mathbf{v} + \mathbf{v}') - I_i(\mathbf{u}_k) \ . \tag{5.27}$$

For homogeneous pixel coordinates of the form $[u_1, u_2, 1]^\top$, the affine warp is then defined as

$$W_{\text{affine}} = \left[\mathtt{A} \,\middle|\, -\mathtt{A}\mathbf{v} + \mathbf{v}'\right] \ . \tag{5.28}$$

As W_{affine} is dependent on the pose perturbation $\boldsymbol{\mu}$ through $\mathbf{v}'$, its derivative is simply the derivative of $\mathbf{v}'$ with respect to $\boldsymbol{\mu}$,

$$\frac{\partial W_{\text{affine}}}{\partial \boldsymbol{\mu}} = \frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \ . \tag{5.29}$$

Eq. (5.11) and Eq. (5.12) become

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right]^\top r_{ijk} \ , \tag{5.30}$$

$$\mathtt{H} = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right]^\top \left[\nabla I_C \frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right] \ . \tag{5.31}$$

Herein lieth one benefit of using the affine warp over the projective warp: $\mathbf{v}'$ does not change from pixel to pixel within the patch. The term may then be factorised out of the inner summation in Eq. (5.30) and Eq. (5.31) become

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right]^\top \left(\sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \right]^\top r_{ijk} \right) \tag{5.32}$$

and

$$\mathtt{H} = \sum_{i} \sum_{j} \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right]^\top \left(\sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C \right]^\top \left[\nabla I_C \right] \right) \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right] \ . \tag{5.33}$$

This factorisation leads to an update step that is computationally more efficient than that in Eq. (5.30) and Eq. (5.31). Rather than a sum of the products of 6 dimensional vectors, the formulation is a sum of the products of 2-dimensional vectors followed by a single left and right multiplication by 6 dimensional vector. It should be noted that the use of the affine warp is particular to our solution which utilises patches. This is due to the fact that the linearisation of the projection becomes inaccurate further from the linearisation point.

5.3.5 Inverse additive formulation with affine warp

The inverse additive formulation [102] for refinement through image alignment is now introduced. This method allows for the derivative of the current image ∇I_C in Eq. (5.11) and Eq. (5.12) to be substituted with the derivative of the patch image ∇I_i. The potential benefits of this are discussed at the end of the section.

Starting with Eq. (5.7)

$$E = \sum_{\mathbf{u}_k \in \Omega_j} \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) + \nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \boldsymbol{\mu} - I_i(\mathbf{u}_k) \right]^2 \tag{5.34}$$

it is now assumed that if $\mathtt{T}_C$ is close to the optimum, the template pixels and the current image

pixels will be approximately equal. That is,

$$I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) \approx I_i(\mathbf{u}_k)\Big|_{\mathbf{u}_k \in \Omega_j} , \tag{5.35}$$

and here, taking the derivatives of both with respect to $\mathbf{u}_k$,

$$\nabla I_C \frac{\partial W}{\partial \mathbf{u}_k} \approx \nabla I_i \Big|_{\mathbf{u}_k \in \Omega_j} . \tag{5.36}$$

Now ∇I_c may be replaced with $\nabla I_i \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1}$ in Eq. (5.34), and the function becomes

$$E = \sum_{\mathbf{u}_k \in \Omega_j} \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) + \nabla I_i \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1} \frac{\partial W}{\partial \boldsymbol{\mu}} \boldsymbol{\mu} - I_i(\mathbf{u}_k) \right]^2 . \tag{5.37}$$

Once again differentiating with respect to $\boldsymbol{\mu}$ and setting to zero:

$$\sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_i \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1} \frac{\partial W}{\partial \boldsymbol{\mu}} \right]^\top \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) + \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1} \frac{\partial W}{\partial \boldsymbol{\mu}} \boldsymbol{\mu} - I_i(\mathbf{u}_k) \right] = 0 . \tag{5.38}$$

Solving for $\boldsymbol{\mu}$ gives

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_i \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1} \frac{\partial W}{\partial \boldsymbol{\mu}} \right]^\top \left[I_C(W(\mathbf{u}_k, \mathtt{T}_C, \mathtt{T}_i)) - I_i(\mathbf{u}_k) \right] , \tag{5.39}$$

where

$$\mathtt{H} = \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_i \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1} \frac{\partial W}{\partial \boldsymbol{\mu}} \right]^\top \left[\nabla I_i \left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1} \frac{\partial W}{\partial \boldsymbol{\mu}} \right] . \tag{5.40}$$

While the benefit of inverse additive methods is that they would allow the pre-computation of the Hessian for each particular patch, this is not the case for re-weighted least-squares. As each term in the Hessian is a function of a new set of weights each iteration, pre-computation is effectively useless. For this reason, the above derivation is in fact a naïve implementation of the inverse additive algorithm: $\mathtt{H}$ still depends on $\boldsymbol{\mu}$ through $\left(\frac{\partial W}{\partial \mathbf{u}_k}\right)^{-1}$ and is thus not constant from iteration to iteration. What the above implementation does allow, however, is for gradient image pixels ∇I_i to be pre-interpolated, potentially leading to a speedup.

5.4 Implementation necessities

5.4.1 Robust estimation

Most estimation problems require some form of insensitivity to outliers in the measurements. M-estimators are used [103] to place more importance on inliers, while down-weighting outliers.

Direct methods are particularly sensitive to outliers. As refinement starts, they have no information about which 3D points are visible or are occluded in the image. Despite this, all 3D points are generally included in the optimisation. Compare this to feature-based methods that have a feature-matching step and may prune landmarks that do not have measurements before refinement begins. Our proposed method uses the Huber loss [103] with a threshold of δ,

$$L_\delta(r_{ijk}) = \begin{cases} \frac{1}{2} r_{ijk}^2 & \text{for } |r_{ijk}| \leq \delta, \\ \delta(|r_{ijk}| - \frac{1}{2}\delta), & \text{otherwise.} \end{cases} . \tag{5.41}$$

The error to minimise during tracking refinement is then:

$$E = \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} L_\delta(r_{ijk}) \ , \tag{5.42}$$

and terms in the update equations are weighted by their respective Huber weights:

$$w(r_{ijk}) = \begin{cases} 1 & \text{for } |r_{ijk}| \leq \delta, \\ \frac{\delta}{r_{ijk}}, & \text{otherwise.} \end{cases} . \tag{5.43}$$

The update equations become:

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \sum_{i \in \mathcal{K}} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} w(r_{ijk}) \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \right] r_{ijk} \ , \tag{5.44}$$

where $\mathtt{H}$ is

$$\mathtt{H} = \sum_i \sum_j \sum_{\mathbf{u}_k \in \Omega_j} w(r_{ijk}) \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \right]^\top \left[\nabla I_C \frac{\partial W}{\partial \boldsymbol{\mu}} \right] \ . \tag{5.45}$$

5.4.2 Coarse-to-fine refinement

To ensure a more global optimum, refinement is done in a coarse to fine manner. Source and target images are downsized a number of times to produce an image pyramid. Tracking refinement begins at the coarsest resolution, and its final estimate is passed to the next stage of refinement. This continues until the original image size is reached. The benefit of this is that at the lowest/coarsest resolution, patches include a greater proportion of the image, increasing the basin of convergence.

5.4.3 Tracking motion model

The cost functions described in Section 5.3 are highly non-convex and requires a good initial estimate of cameras pose before refinement. For this reason, a motion model is utilised to provide an initial estimate of the current camera pose given the previous one.

We use the decaying-velocity motion model. Given the previous pose of the camera $\mathtt{T}_{prev}$, the current estimate of the pose is calculated as:

$$\mathtt{T}_C = \exp(\boldsymbol{\mu}_{MM})\mathtt{T}_{prev} \ , \tag{5.46}$$

where $\boldsymbol{\mu}_{MM}$ is the velocity estimate provided by the motion model. Once refinement has finished, the model is updated. The velocity between the previous pose and the current pose is measured as:

$$\boldsymbol{\mu}_{vel} = \log_{SE(3)}(\mathtt{T}_C\mathtt{T}_{prev}^{-1}) \ , \tag{5.47}$$

where $\log_{SE(3)}()$ is the matrix logarithm for $\mathbf{SE}(3)$, the inverse of the matrix exponential $\exp()$. The motion model is then updated as:

$$\boldsymbol{\mu}_{mm} \leftarrow \lambda(\frac{1}{2}\boldsymbol{\mu}_{vel} + \frac{1}{2}\boldsymbol{\mu}_{MM}) \ , \tag{5.48}$$

where λ is a velocity-decay parameter, set to around 0.9.

5.5 Experiments

A camera tracker for direct SLAM has been introduced that uses sparse corners in the image rather than regions of high image gradient. This section presents a qualitative evaluation of the accuracy

and the computational efficiency of the tracker compared to a current state of the art sparsified direct method, namely LSD-SLAM [16].

We use the TUM dataset [104] for evaluation, which provides video sequences from an RGB-D device. Ground-truth camera poses are also included. Both our method and LSD-SLAM require depth information to function. Our method requires knowledge of the associated depth of corners, and LSD-SLAM requires depth of high-gradient regions in the image. For this reason, the depth images provided by the TUM-dataset will be used to provide this information. At the first frame of each sequence a new keyframe is created and tracking is then allowed to proceed. Keyframes are added to the map at fixed intervals that are the same for both trackers. When a keyframe is added, either corners or high-gradient regions are extracted from the image and assigned depths from the appropriate depth image.

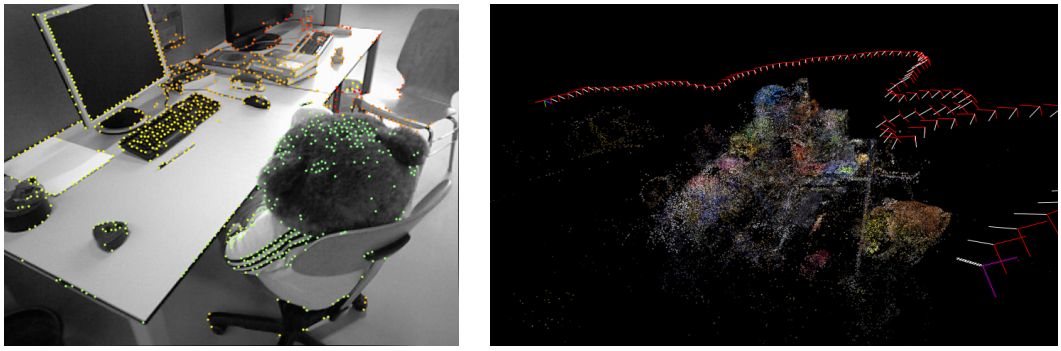
In Section 5.5.1 our corner-based method is compared to LSD-SLAM. All of the proposed warps from Section 5.3 are used and compared. Initially a single reference keyframe is used for all methods. In Section 5.5.2 the computational cost of all methods is compared. This information provides insight into the overhead that is allowed when tracking using multiple reference keyframes. Following this, Section 5.5.3 provides a comparison between the proposed tracker with a single reference frame, and the same tracker with multiple reference frames. A benefit of direct methods is their ability to maintain accuracy when the image is blurred. Section 5.5.4 therefore compares our method and LSD-SLAM when tracking using a blurry image.

5.5.1 Results: single keyframe

Recall that our proposed method is able to minimise photometric error across patches from *multiple* reference keyframes in the map. For this section, we constrain our proposed tracker to a *single* reference keyframe in order to compare it to LSD-SLAM, which also uses a single keyframe.

Around 1000 corners are extracted from each keyframe. Fig. 5.2(a) shows an example of these corners from a reference keyframe projected into the current frame. Fig. 5.2(b) shows the resultant 3D point-cloud when the corners in keyframes are provided depth from the relevant depth images.

Table 5.1 shows absolute trajectory error for a number of estimated trajectories using the proposed method. The error when using our method with the projective warp, the affine warp, the inverse additive formulation are compared to the error when using LSD-SLAM for camera tracking.



(a) (b)

Figure 5.2: (a) Example tracking frame from the TUM dataset [104] and (b) map used for the proposed camera tracking method. Map structure was obtained using a depth sensor and consists of sparse 3D points from corners with associated depth.

Examination of the results shows that patch-based methods have comparable, if not better results, than those of LSD-SLAM. This shows that corners provide more than enough image information for reliable camera tracking. The proposed affine warp, although a linear approximation to the projective warp, had almost equal accuracy. For the long_office-_household sequence, it actually performed slightly *better* than the projective warp. The reasons for this are unclear, however we postulate that the linearisation may lead to better convergence in some cases.

Of all three corner-based methods, the inverse additive formulation performed the worst. This is because it uses an approximation to the gradient of error function which only holds near the optimum solution.

	Absolute Error [cm]			
Sequence	Projective	Affine	AffineInv	LSD-SLAM
freiburg2_xyz	1.8	1.8	1.8	**1.6**
freiburg2_desk	**9.7**	10.8	19.3	12.9
freiburg3_long_office_household	10.9	**8.8**	14.7	9.3

Table 5.1: Absolute trajectory error of proposed method using different warps vs LSD-SLAM [16]. All methods use a single reference keyframe.

For qualitative evaluation, Fig 5.3 shows the estimated trajectories for all three sequences using the affine warp. While the estimated trajectories match ground-truth very well, the longer trajectories (freiburg2_desk and freiburg3_long_office_household) do accumulate rotational and translational drift over time, leading to a less precise trajectory and greater error. As the sequences have loops, accurate loop closure would decrease error further.

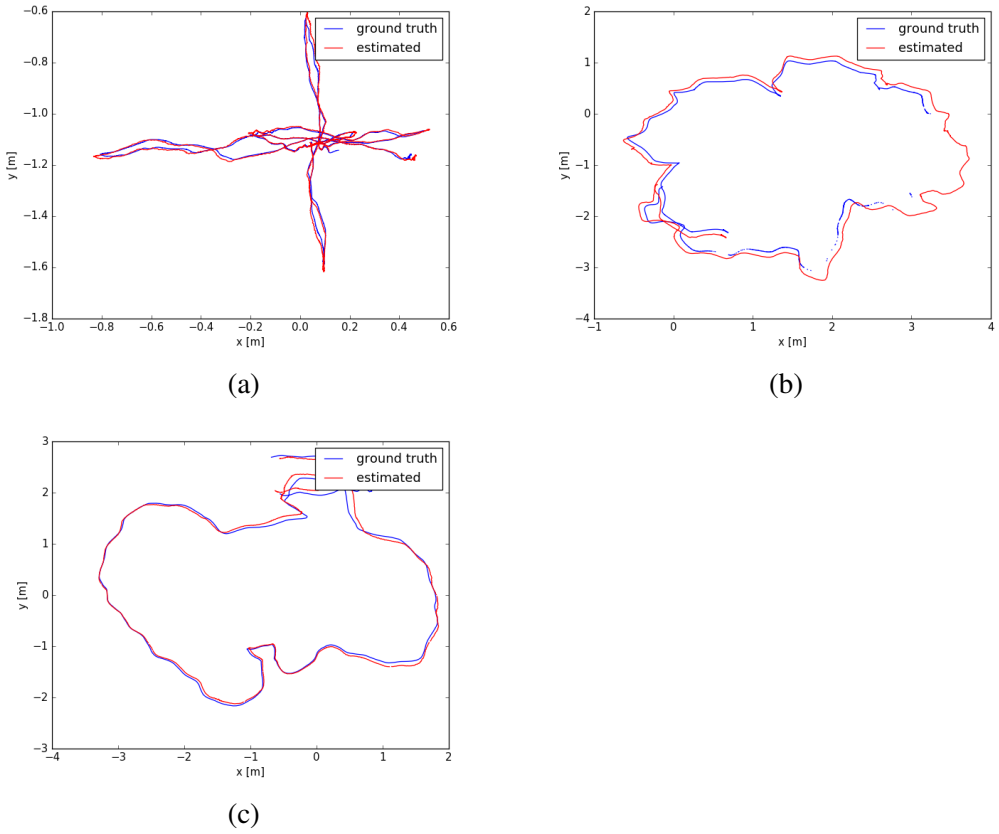


Figure 5.3: Tracking results for (a) freiburg2_xyz (b) freiburg2_desk and (c) freiburg3_long_office_household using the proposed affine tracker. Estimated trajectory is shown in red and ground truth in blue. The estimated trajectories match ground-truth almost perfectly, although longer trajectories are more erroneous due to drift.

5.5.2 Results: computational efficiency

Section 5.5.1 has shown that utilising corners rather than high-gradient regions allows for reliable tracking accuracy. In this section, the speed of each method is compared. Tests were run using an Intel i7-4770K CPU running at 3.5GHz. LSD-SLAM is able to make use of streaming SIMD extensions to perform some computations in parallel. Although it is possible for the proposed tracking method to make use of SIMD extensions, they have been disabled for the purposes of comparison. Fig 5.4. shows the time spent tracking for each method on each of the test sequences.

While LSD-SLAM spends on average 40ms per frame, all of the proposed formulations use a constant 20ms per frame.

It should be noted that specifically for *tracking* a significant pre-computation may be made for the projective warp. The 3D position $\mathbf{X}_{i,j,k}$ of each source pixel in Eq. (5.14) may be pre-computed as it is not a function of the current camera pose T_C.

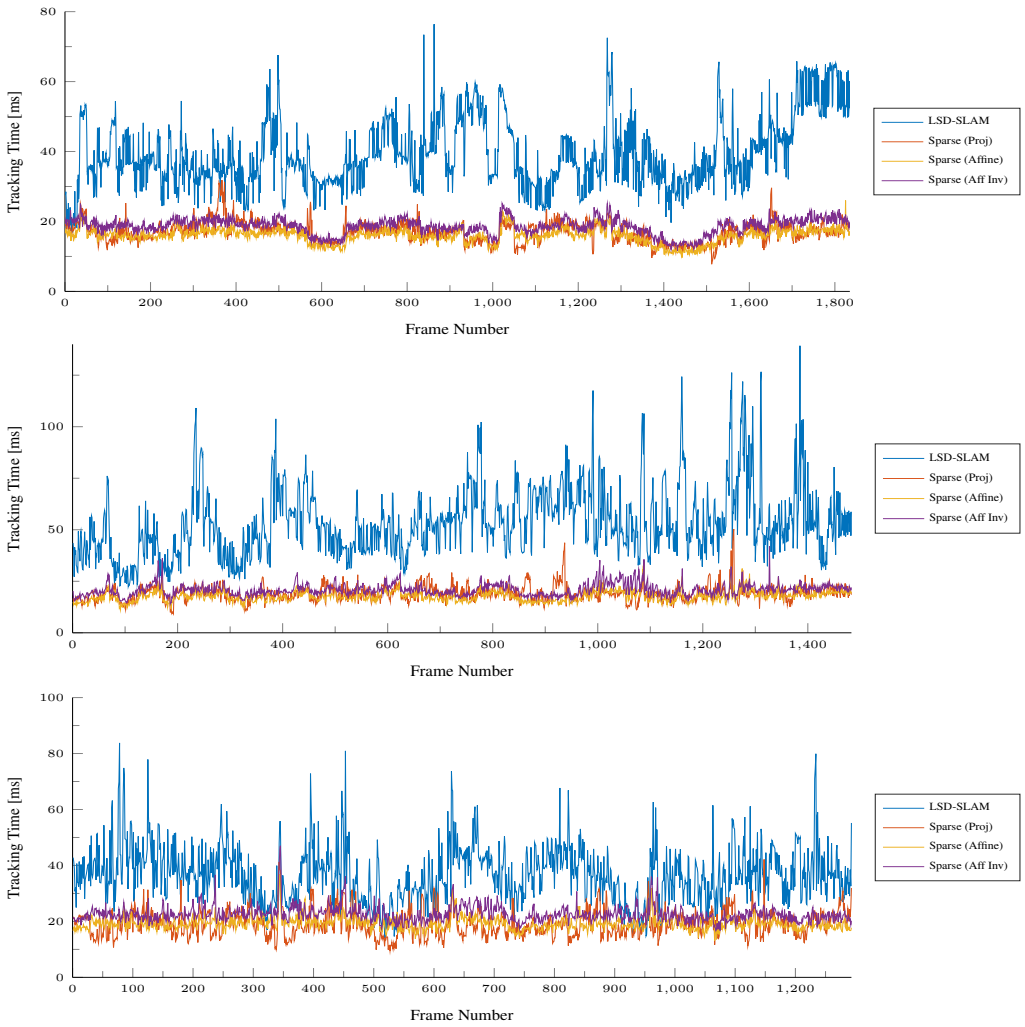


Figure 5.4: Tracking times for the freiburg2_xyz sequence (top), freiburg2_desk sequence (middle), and freiburg3_long_office_household sequence (bottom).

Surprisingly the inverse additive formulation is slower than its forward counterpart. This suggests that the additional warp matrix inversions required are more costly than bi-linear interpolation of gradients of the tracking image. As it offers worse performance as far as accuracy and speed are concerned, the inverse affine formulation is abandoned in favour of its forward counterpart for the remainder of this section.

5.5.3 Results: use of multiple reference keyframes

Tracking using multiple reference keyframes is now introduced. Section 5.5.2 has shown that the proposed method takes approximately 20ms per frame using 1000 corners, and it is safe to assume that tracking can make use of a total of 2000 corners while maintaining tracking rates of around

25Hz.

An interesting question to ask is whether using multiple reference keyframes leads to an increase in accuracy. We now choose to split these 2000 points across a set of surrounding keyframes that will be used as reference frames. We use 10 keyframes where each keyframe is allowed to detect 200 corners. Fig. 5.5 shows an overview of how tracking using multiple reference keyframes takes place. Table 5.2 shows the absolute trajectory error for the sparse affine method using a single reference keyframe with 1000 and a set of 10 reference keyframes each with 200 corners.

	Absolute Error [cm]	
Sequence	1 keyframe	10 keyframes
freiburg2_xyz	1.8	**1.7**
freiburg2_desk	**10.8**	44.6
freiburg3_long_office_household	8.8	**8.6**

Table 5.2: Absolute trajectory error of the proposed method using an affine warp using a single reference keyframe vs the same method using 10 reference keyframes.

While splitting patches across keyframes does seem to slightly decrease error in two of the sequences, it results in a marked *increase* in error for the freiburg2_desk sequence. A highly probable cause of this error is a problem in general with direct methods: *it is difficult to deal with occlusions*. While feature-based methods are able to tell whether a point is visible in the current frame through feature-matching, direct methods have to include all possible points in the optimisation, and rely on a robust estimator to ignore occluded ones. The freiburg2_desk sequences includes a desk with a number of objects which occlude each other frequently throughout the sequence. Including more keyframes further from the current camera estimate is highly likely to result in their 3D points being occluded in the current frame. A potential solution to this might be to impose a pose-difference threshold between a keyframe and the current frame. Keyframes

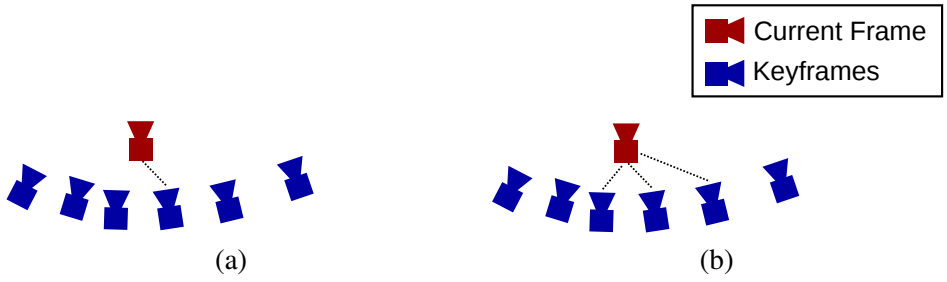
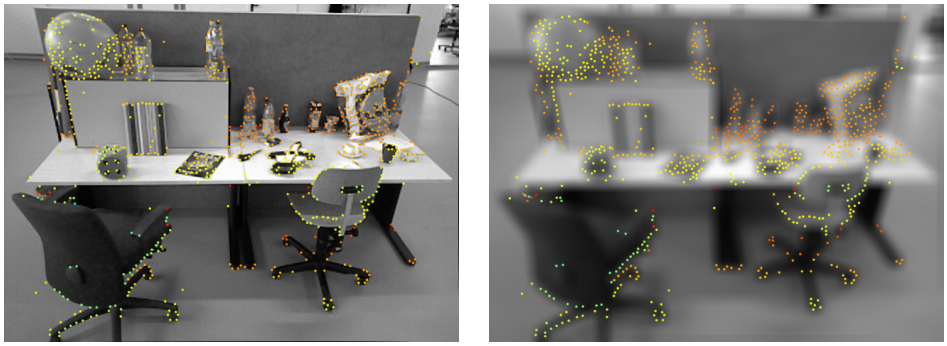


Figure 5.5: (a) Tracking using a single reference keyframe and (b) tracking using multiple reference keyframes. Black dashed lines indicate a keyframes back-projection and reprojection of its patches in the current frames image as shown in Fig. 5.1



(a) (b)

Figure 5.6: (a) Example tracking frame and (b) its corresponding blurred image

above this threshold would not be allowed as reference frames. While it may seem possible that the lower number of corners in each frame were to blame for the larger error, evaluation with a single reference frame using 200 corners resulted in an error of 9.2cm, showing this is not the case.

5.5.4 Results: blurry image

An advantage of direct methods over feature-based ones is their invariance to motion blur and defocus in the image. For this reason, we compare the performance of our proposed method with LSD-SLAM when placed under such circumstances.

We postulate that a possible benefit of using multiple reference frames is that the additional image information from multiple views may add additional redundancy which aids tracking when the image is blurred.

To simulate motion blur or defocus, both LSD-SLAM's tracker and the 10-keyframe affine tracker are run on sequences that are blurred using a normalised box filter of 20×20 pixels in size. Note that as this is to simulate difficult *tracking*, images used for keyframes are *not* blurred. Fig. 5.6 shows the result of a tracking frame before and after being blurred.

Table 5.3 shows the absolute error in the estimated trajectories from both LSD-SLAM and our 10-keyframe affine tracker when tracking under constant image blur. For each sequence the multi-keyframe method outperforms LSD-SLAM suggesting that multiple view angles do increase tracking accuracy when the image is blurred.

	Absolute Error [cm]	
Sequence	Affine (10 keyframes)	LSD-SLAM
freiburg2_xyz	**4.1**	10.1
freiburg2_desk	**48.4**	54.7
freiburg3_long_office_household	**50.5**	53.9

Table 5.3: Absolute trajectory error of the proposed method using 10 reference frames and LSD-slam [16] while tracking under heavy blur.

5.6 Conclusion

In this chapter a direct camera tracker has been introduced that operates on patches around a sparse set of corners in the image rather than the whole image or high image-gradient regions.

We have presented methods of warping patches from a reference keyframe to the current image frame in order align those patches to the current image. A novel affine warp that may only be used with sparse corners was introduced. There are two benefits to using this warp over the more widely used projective warp. Firstly for each patch only the centre pixel needs to be back-projected and projected into the current frame. All other pixel locations may be computed using the affine warp. The second is that update equations for pose refinement factorise easily, reducing computational cost. We have shown that the proposed method obtains results that are almost equal if not better in some cases to a method that uses high image-gradient regions in the image. As computational cost is reduced, the proposed method only spends 20ms per frame on pose refinement, which is half that of the high image-gradient method. This reduced computational cost allows for multiple reference keyframes to be used for tracking. Although using multiple reference keyframes does increase accuracy in some cases, occlusion is more likely due to the larger pose differences between reference keyframes and the current frame. In these cases accuracy is severely degraded.

As occlusion remains a problem, future research would be to accurately model which patches are in the image before optimisation. A potential solution would be to continually monitor the photometric error of patches over time. If a patch's error suddenly increased and remained high for an extended period of time, it would be considered "lost" and no longer used in tracking.

Finally, we note that that very recently, Engel *et al.* have published a sparsified version of their LSD-SLAM system [36], showing, in findings similar to ours, that sparse corners are more accurate than high image-gradient regions.

Chapter 6

Sparsified direct mapping

This chapter introduces a sparsified direct map refinement algorithm that uses bundle adjustment rather than depth filtering. The structure estimated by the method is compared to that estimated by filtering, where it often achieves more accurate results, despite having more trouble with occlusion.

6.1 Introduction

The previous chapter developed a camera pose tracker that operates through minimisation of *photometric* error rather than geometric error. This chapter is a continuation of that work, introducing a map refinement algorithm that operates through minimisation of photometric error.

Owing to the computational cost associated with a large number of pixels, most direct methods utilise a form of filtering for structural estimation [16, 34] rather than a least-squares refinement. For that filtering, a large number of small-baseline measurements are used in preference to fewer large-baseline ones.

In this chapter, we introduce a least-squares refinement, preferring sparser large-baseline measurements. We discuss how the affine approximation introduced in the previous chapter significantly speeds up computation time when refining structure. Filtering-based techniques only optimise depths within a keyframe, and therefore often require some form of pose-graph optimisation for refinement between them. The benefit of the proposed method is a *joint* refinement on both keyframes and structure.

In Section 6.2 current direct map-refinement methods are discussed. These include filtering-based methods, which are popular for real-time use; and least-squares methods, which are generally limited to structure from motion and multi-view stereo algorithms. In Section 6.3 the proposed

method for sparse photometric structural refinement is introduced. This includes the formulation of the optimisation, the choice of warp between keyframes, and methodology for solving the optimisation itself. For comparative purposes, Section 6.4 reviews a popular filtering method. Section 6.5 introduces details for a full SLAM method that operates through direct tracking and structural estimation using the proposed method. In Section 6.6 the refinement is evaluated. The least-squares refinement is compared to filtering-based methods in the context of structural estimation and operation in low frame-rate video sequences. Quantitative results are also included and discussed. Finally, concluding remarks are presented in Section 6.7.

6.2 Related work

In this chapter a sparse bundle adjustment is introduced that operates through reduction of photometric error rather than geometric error. In essence, the objective of this work is similar to that of Klein and Murray [9]. At the time, filtering based-methods were the de facto standard for real-time monocular SLAM, whereas structure from motion methods were for offline use. Similarly today, CPU-based direct methods utilise filtering for depth estimation, while offline multi-view stereo algorithms often use a photometric error-term at some stage of their pipelines. For this reason, this section discusses and compares filtering-based and bundle adjustment-based direct methods in the context of the proposed method.

Photometric terms have often been used in multi-view stereo, a field that focuses on the estimating structure using a set of fixed cameras. An example is the work by Furukawa and Ponce [105] who used a photometric error term to produce a dense set of patches from images. In addition to estimating patches 3D position, their orientation is also modelled. Rather than relying solely on a robust error function to deal with occlusion, the method models visibility information along with geometric information. Their work is later extended to estimate both camera parameters in addition to geometric structure [106].

A more recent work is the photometric bundle adjustment proposed by Delaunoy and Pollefeys [107], who model structure of the scene as a mesh rather than a set of patches. As visibility information is intrinsic in this formulation, it needn't be modelled independently. In addition to structural information, the texture of the mesh is modelled and refined. Like [106], camera calibration is estimated alongside geometric information. DTAM by Newcome *et al.* [14] also

estimates dense structure using a photometric cost. As the method is for real-time use, a GPU is used for computation.

CPU-based direct approaches utilise probabilistic methods as a real-time replacement to bundle adjustment. Early works in this field include Matthies and coworkers' incremental depth estimation method [108] that uses a Kalman filter to estimate depth from stereo pairs. Here the Kalman filter's system state models the inverse depth of each pixel in an image of interest. Inverse depth is chosen as it is directly proportional to the disparity delivered from an optical flow algorithm. Additionally it is better conditioned for more distant objects, and its relationship with disparity allows estimating variance to set search limits on the matching algorithm. This type of filtering is used for depth estimation in the semi-dense odometry method by Engel *et al.* [35] and in the LSD-SLAM method [16].

Forster and coworkers' Semi-Direct Visual Odometry method [34] also utilises depth-based filtering for structural estimation. While the structural estimation is in fact direct, operating through image patch refinement, the final structural refinement is feature-based. Once a frame has been localised, patches are independently refined, producing a set of reprojection errors that are finally minimised using a bundle adjustment. The filtering technique used is a modified version of a method introduced by Vogiatzis and Hernández [32], which was devised originally for stereo depth estimation. It assumes that depth measurements are drawn from a mixture model of a Gaussian distribution for "true" measurements and a uniform distribution for outliers. Filtering estimates both the true depth of the Gaussian distribution, and the mixture parameter which represents the probability of a measurement being an outlier. While the initial work estimates depth, Forster *et al.* modify it to estimate *inverse* depth, as this is shown to be more Gaussian distributed than depth itself [109].

The method proposed here is a mid-point between filtering-based solutions to direct refinement and dense bundle adjustment. Like multi-view stereo a photometric bundle adjustment is used, however sparse corners are estimated rather than a dense surface model. While offline methods are able to model patch orientation, view information, and texture; the proposed bundle adjustment must be used in a real-time method. It therefore relies on affine-warped rather than oriented patches for estimated keyframes. It does not model view information, but relies on a robust estimator to down-weight unseen patches.

6.3 Map refinement using direct image alignment

The proposed method of direct map refinement using least squares is now set out. This section presents a sparse bundle adjustment that operates through enforcing photometric consistency rather than minimising reprojection error. In Section 6.3.1 the photometric error function to be minimised is introduced. Like tracking, the proposed optimisation requires a choice of warp from one keyframe to another. Section 6.3.2 shows how the use of an affine warp is almost essential in the proposed optimisation. Section 6.3.3 shows how the optimisation is solved using a Levenberg-Marquardt scheme. Like tracking, the proposed solution requires a robust estimator to deal with occlusion. Details of this are discussed in Section 6.3.4.

6.3.1 Optimisation formulation

We shall first start with a simple problem of two keyframe refinement and then generalise this to multiple frames. The formulation is similar to the direct tracking refinement used in Chapter 5 Section 5.3.2.

Given two keyframes with poses $\mathtt{T}_{i1}$ and $\mathtt{T}_{i2}$ and images $I_{i1}()$ and $I_{i2}()$, we first assume that only the keyframe $\mathtt{T}_{i1}$ has a set of corners $\mathbf{v}_j \in \mathcal{C}_{i1}$ that have estimated depths $d_j \in \mathcal{D}_{i1}$. Minimising the following function with respect the pose $\mathtt{T}_{i2}$ is equivalent to the direct tracking problem introduced in the previous chapter:

$$\mathtt{T}_{i2} = \arg\min_{\mathtt{T}_{i2}} \sum_{j \in \mathcal{C}_{i1}} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_{i2}(W(\mathbf{u}_k, d_j, \mathtt{T}_{i2}, \mathtt{T}_{i1})) - I_{i1}(\mathbf{u}_k) \right]^2 , \tag{6.1}$$

where $\mathbf{u}_k$ is a pixel in a set of pixels Ω_j belonging to the patch associated with corner $\mathbf{v}_j$. We note that this refinement takes place by finding the minimally parameterised camera *offset* $\boldsymbol{\mu} \in \mathfrak{se}(3)$. However for the sake of simplicity this will be implied rather than explicitly stated from now on. As inverse depths are more normally distributed than depths, we use the following parameterisation:

$$\rho_j = 1/d_j \tag{6.2}$$

where ρ is the inverse depth that is stored and refined. For map refinement, the keyframe $\mathtt{T}_{i1}$ and the inverse depths $\mathcal{P}_{i1}$ of its corners are no longer fixed, but free parameters that are estimated as well. Now the refinement is to find the poses $\mathtt{T}_{i1}$ and $\mathtt{T}_{i2}$ and the inverse depths $\mathcal{P}_{i1}$ that minimise

Eq. (6.1):

$$\mathtt{T}_{i1}, \mathtt{T}_{i2}, \mathcal{P}_{i1} = \underset{\mathtt{T}_{i1}, \mathtt{T}_{i2}, \mathcal{P}_{i1}}{\arg\min} \sum_{j \in \mathcal{C}_{i1}} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_{i2}(W(\mathbf{u}_k, \rho_j, \mathtt{T}_{i2}, \mathtt{T}_{i1})) - I_{i1}(\mathbf{u}_k) \right]^2 . \tag{6.3}$$

This formulation, however, ignores the fact that keyframe $\mathtt{T}_{i2}$ has its own set of corners $\mathcal{C}_{i2}$ with associated inverse depths $\mathcal{P}_{i2}$. The final symmetric refinement then is to find the poses $\mathtt{T}_{i1}$ and $\mathtt{T}_{i2}$ and the inverse depths $\mathcal{P}_{i1}, \mathcal{P}_{i2}$ that minimise:

$$\mathtt{T}_{i1}, \mathtt{T}_{i2}, \mathcal{P}_{i1}, \mathcal{P}_{i2} =$$
$$\underset{\mathtt{T}_{i1}, \mathtt{T}_{i2}, \mathcal{P}_{i1}, \mathcal{P}_{i2}}{\arg\min} \sum_{s,t \in \{i1,i2\}, s \neq t} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_t(W(\mathbf{u}_k, \rho_j, \mathtt{T}_t, \mathtt{T}_s)) - I_s(\mathbf{u}_k) \right]^2 . \tag{6.4}$$

In this formulation, the keyframes $\mathtt{T}_s$ and $\mathtt{T}_t$ shall be referred to as the *source* keyframe and *target* keyframe respectively. Note that the nomenclature is only dependent on where the inverse depths $\mathcal{P}$ originate: *i.e.* a keyframe can be a source and a target in the same optimisation.

We now extend the optimisation to multiple keyframes. Consider a set of keyframes $\mathcal{K}$ each with a set of corners $\mathcal{C}_i$ with inverse depths $\mathcal{P}_i$. The goal is then to find a set of keyframe poses $\mathcal{K}$ and inverse depths $\mathcal{P} = \{\mathcal{P}_i, \mathcal{P}_2, ..., \mathcal{P}_N\}$ that minimise:

$$\mathcal{K}, \mathcal{P} = \underset{\mathcal{K}, \mathcal{P}}{\arg\min} \sum_{s,t \in \mathcal{K}, s \neq t} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_t(W(\mathbf{u}_k, \rho_j, \mathtt{T}_t, \mathtt{T}_s)) - I_s(\mathbf{u}_k) \right]^2 \tag{6.5}$$

$$= \underset{\mathcal{K}, \mathcal{P}}{\arg\min} \sum_{i,h \in \mathcal{K}, s \neq t} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} r_{stjk}^2 \tag{6.6}$$

6.3.2 Choice of warp

As Eq. (6.5) is solved for sparse corners in the source keyframe, the affine warp discussed in Chapter 5 is used:

$$W_{\text{affine}} = \left[\mathtt{A} \middle| -\mathtt{A}\mathbf{v} + \mathbf{v}' \right] , \tag{6.7}$$

where $\mathtt{A}$ is a first order derivative of the projective warp of the corner $\mathbf{v}$, and $\mathbf{v}'$ is the projective warp of $\mathbf{v}$ from the source keyframe to the target keyframe. For the proposed direct bundle adjustment, the affine warp has a *significant* benefit over the projective warp. Recall from Chapter 5 that

the projective warp back-projects the pixel $\mathbf{u}_k$ in a patch into the 3D point $\mathbf{X}_{i,j,k}$. For tracking, the set of 3D points could be pre-computed before optimisation leading to a significant speed up. This pre-computation is impossible using projective warping for the proposed bundle adjustment as the depth that is used for $\mathbf{X}_{i,j,k}$ is a free parameter. While the pre-computation is also impossible for an affine warp, the back-projection need only be computed *once* per patch for its centre, the corner $\mathbf{v}$.

6.3.3 Optimisation solution

Each pixel difference r_{stjk} is lumped into the vector $\mathbf{r}$, and the parameters for keyframes and inverse depths into the vector $\boldsymbol{x}$. Eq. (6.5) becomes:

$$\boldsymbol{x} = \arg\min_{\boldsymbol{x}} \sum \mathbf{r}^\top \mathbf{r} \,. \tag{6.8}$$

This is the same form of as a feature-based bundle adjustment. Solving Eq. (6.8) is done in an iterative manner using Levenberg-Marquardt:

$$(\mathtt{H} + \lambda \mathsf{diag}(\mathtt{H}))\delta\boldsymbol{x} = -\mathbf{g} \,, \tag{6.9}$$

where $\mathtt{H} = \mathtt{J}^\top\mathtt{J}$ is the least-squares approximation to the Hessian matrix and $\mathbf{g} = \mathtt{J}^\top\mathbf{r}$ is the gradient vector. Fig. 6.1 shows the structure of the Hessian and gradient vector for the proposed direct bundle adjustment.

- Red terms correspond to offsets to keyframes. For the keyframe $\mathtt{T}_i$ we define the 6×6 red block in the Hessian for each keyframe as $\mathtt{H}_i$ and the 6×1 vector in the gradient vector as $\mathbf{g}_i$.
- Blue terms are associated with inverse depths. For the inverse depth ρ_j we define the corresponding singe element in the Hessian as h_j and the single element in the gradient vector as g_j.
- Green blocks correspond to photometric errors between inverse depths and keyframes. These blocks are densely populated as the photometric error for an inverse depth is evaluated at *every* other keyframe regardless of whether it is actually visible in it or not. One exception

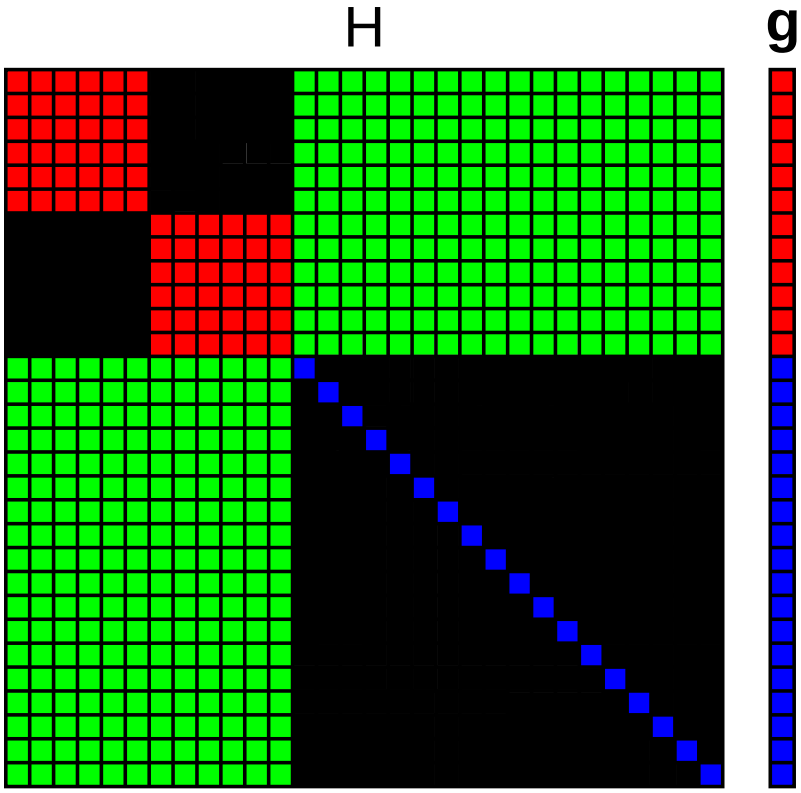


Figure 6.1: Hessian and gradient vector structure for the proposed method. See text for key.

to this is if the corner $\mathbf{v}'$ does not lie in the image plane. As the pose of a source keyframe affects photometric residuals, constraints additionally exist between a inverse depth and its source keyframe. We define the 6×1 constraint between keyframe $\mathtt{T}_i$ and inverse depth ρ_j in the upper part of the Hessian as $\mathtt{W}_{ij}$. As the Hessian is symmetric, the terms in its bottom left part are equal to $\mathtt{W}_{ij}^\top$.

Recall from Chapter 5 that the affine warp allows for factorisation of update equations for the tracking frame. For a single patch, the update equations for the tracking frame are:

$$\boldsymbol{\mu} = \mathtt{H}^{-1} \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right]^\top \left(\sum_{\mathbf{u}_k \in \Omega_j} [\nabla I_C]^\top r_{stjk} \right) , \tag{6.10}$$

where

$$\mathtt{H} = \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right]^\top \left(\sum_{\mathbf{u}_k \in \Omega_j} [\nabla I_C]^\top [\nabla I_C] \right) \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}} \right] . \tag{6.11}$$

The summation terms in Eq. (6.10) and Eq. (6.11) are equivalent to the gradient and Hessian matrix of the total photometric error for the patch with respect to the patches centre in the target

keyframe $\mathbf{v}'$. Therefore we define:

$$\mathbf{g}_{\text{patch}} = \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C\right]^\top r_{stjk} \tag{6.12}$$

$$\mathtt{H}_{\text{patch}} = \sum_{\mathbf{u}_k \in \Omega_j} \left[\nabla I_C\right]^\top \left[\nabla I_C\right] \ . \tag{6.13}$$

For each patch in a source and target pair, the updates to the Hessian and gradient terms are as follows. If the keyframe is a target keyframe:

$$\mathbf{g}_i = \mathbf{g}_i + \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_t}\right]^\top \mathbf{g}_{\text{patch}} \tag{6.14}$$

$$\mathtt{H}_i = \mathtt{H}_i + \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_t}\right]^\top \mathtt{H}_{\text{patch}} \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_t}\right] \ , \tag{6.15}$$

where $\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_t}$ is the equivalent to the $\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}}$ in Eq. (6.10) and Eq. (6.11).

If the keyframe is a source keyframe:

$$\mathbf{g}_i = \mathbf{g}_i + \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_s}\right]^\top \mathbf{g}_{\text{patch}} \tag{6.16}$$

$$\mathtt{H}_i = \mathtt{H}_i + \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_s}\right]^\top \mathtt{H}_{\text{patch}} \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_s}\right] \ . \tag{6.17}$$

For inverse depths:

$$g_j = g_j + \left[\frac{\partial \mathbf{v}'}{\partial d_j}\right]^\top \mathbf{g}_{\text{patch}} \tag{6.18}$$

$$h_j = h_j + \left[\frac{\partial \mathbf{v}'}{\partial d_j}\right]^\top \mathtt{H}_{\text{patch}} \left[\frac{\partial \mathbf{v}'}{\partial d_j}\right] \ , \tag{6.19}$$

where the terms $\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_s}$ and $\frac{\partial \mathbf{v}'}{\partial d_j}$ are the derivatives of the projection of the corner $\mathbf{v}$ in the target keyframe $\mathtt{T}_t$ with respect the a change in the source keyframe $\mathtt{T}_s$ and the corners inverse depth ρ_j respectively:

$$\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_s} = \mathtt{J}_{\text{projpoint}} \mathtt{J}_{\text{IDsource}} \tag{6.20}$$

$$\frac{\partial \mathbf{v}'}{\partial d_j} = \mathtt{J}_{\text{projpoint}} \mathtt{J}_{\text{IDDepth}} \ . \tag{6.21}$$

Recall that the inverse projection of the corner produces a 3D point, which is subsequently projected into the target keyframe. Here $\mathtt{J}_{\text{projpoint}}$ is the derivative of the projection $\mathbf{v}'$ with respect to that 3D points position. The points position is a function of inverse depth ρ_j and so $\mathtt{J}_{\text{IDDepth}}$ is the derivative of the 3D point with respect to the inverse depth. The 3D point is also a function of the source keyframes pose, and thus $\mathtt{J}_{\text{IDsource}}$ is the derivative of the point with respect to the source pose. Details are found in Appendix A.

Finally, the keyframe to inverse depth constraints $\mathtt{W}_{ij}$ are updated as follows. If $\mathtt{T}_i$ is a target keyframe

$$\mathtt{W}_{ij} = \mathtt{W}_{ij} + \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_t}\right]^\top \mathtt{H}_{\text{patch}} \left[\frac{\partial \mathbf{v}'}{\partial d_j}\right] , \tag{6.22}$$

but if $\mathtt{T}_i$ is a source keyframe

$$\mathtt{W}_{ij} = \mathtt{W}_{ij} + \left[\frac{\partial \mathbf{v}'}{\partial \boldsymbol{\mu}_s}\right]^\top \mathtt{H}_{\text{patch}} \left[\frac{\partial \mathbf{v}'}{\partial d_j}\right] . \tag{6.23}$$

6.3.4 Robust estimator

As with tracking, a robust estimator must be used to account for large residuals due to occlusion. Adding the Huber norm to Eq. (6.5) results in:

$$\mathcal{K}, \mathcal{P} = \underset{\mathcal{K},\mathcal{P}}{\arg\min} \sum_{i,h \in \mathcal{K}, s \neq t} \sum_{j \in \mathcal{C}_i} \sum_{\mathbf{u}_k \in \Omega_j} L_\delta(r^2_{stjk}) \tag{6.24}$$

Using this formulation simply requires weighting patch gradient and Hessian terms of the form:

$$\mathbf{g}_{\text{patch}} = \sum_{\mathbf{u}_k \in \Omega_j} w(r_{stjk}) \left[\nabla I_C\right]^\top r_{stjk} , \tag{6.25}$$

$$\mathtt{H}_{\text{patch}} = \sum_{\mathbf{u}_k \in \Omega_j} w(r_{stjk}) \left[\nabla I_C\right]^\top \left[\nabla I_C\right] , \tag{6.26}$$

where $w(r_{stjk})$ is the Huber weight defined in the previous chapter.

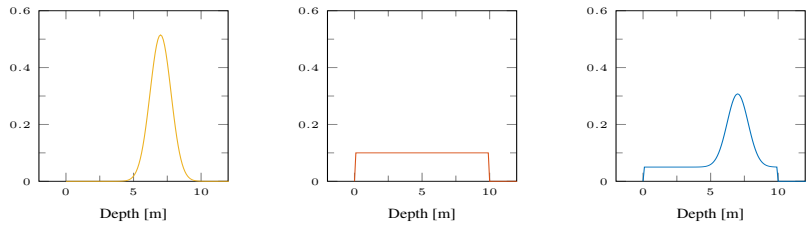


Figure 6.2: Probability distributions for depth assuming the depth measurement is an inlier (left) or an outlier (centre). The rightmost distribution is the probability of any depth measurement. The inlier and outlier distributions are combined using an inlier/outlier ratio.

6.4 Depth filtering

To provide context for the proposed bundle adjustment, an overview of a popular depth-filtering method is presented. This section details a filtering method introduced by Vogiatzis and Hernández [32] for incorporating a large number of noisy depth measurements into a single estimate. Although compatible with an inverse-depth formulation, the method in [32] uses depth estimates.

We refer to Fig.5.1 of the previous chapter. There, a keyframe with known pose and corners with known associated depths was used to estimate the pose of the current frame. Here, the poses of *both* frames are considered known, and the depths in the keyframe are uncertain. Measurements for uncertain depths are obtained by using the following refinement:

$$\mathcal{D}_{i1} = \arg\min_{\mathcal{D}_{i1}} \sum_{j \in \mathcal{C}_{i1}} \sum_{\mathbf{u}_k \in \Omega_j} \left[I_{i2}(W(\mathbf{u}_k, d_j, \mathtt{T}_{i2}, \mathtt{T}_{i1})) - I_{i1}(\mathbf{u}_k) \right]^2 . \tag{6.27}$$

The method for updating probabilistic depth estimates is now described. It is assumed that that measured depth is drawn from a mixture of distributions as shown in Fig. 6.2. Inlier depth estimates are assumed to be drawn from a Gaussian distribution centred around the true estimate, while outliers are considered to be drawn from a uniform distribution between a maximum and minimum allowed set of depths.

The probability of a depth measurement d is given by:

$$p(d|Z, \pi) = \pi \mathcal{N}(d|Z, \sigma_Z^2) + (1 - \pi)\mathcal{U}(d|Z_{min}, Z_{max}) , \tag{6.28}$$

where Z is the "true" depth, and σ_Z^2 is the variance of its measurement. The uniform distribution allows a maximum Z_{max} and minimum Z_{min} depth. The parameter π controls the probability of

the depth measurement being an inlier.

Given a set of depth measurements $d_1, d_2, ..., d_N$, the likelihood of the true depth Z and inlier probability π is proportional to the probability of the measurements:

$$p(Z, \pi | d_1, d_2, ..., d_N) \propto p(Z, \pi) p(d_1, d_2, ..., d_N | Z, \pi) \ , \tag{6.29}$$

where $p(Z, \pi)$ is a prior on the depth and inlier ratio. As each measurement is considered independent, Eq. (6.29) becomes

$$p(d_1, d_2, ..., d_N | Z, \pi) = p(Z, \pi) \prod_{i=1}^{N} p(d_i | Z, \pi) \ . \tag{6.30}$$

To model Z and π, their values may be quantised to produce a two dimensional histogram, which is updated at each depth measurement. For a large amount of corners in the image, however, this becomes infeasible. For this reason, Vogiatzis and Hernández introduce a parametric model. The posterior is modelled as follows:

$$q(Z, \pi | a_n, b_n, \mu_n, \sigma_n) = \mathrm{B}(\pi | a_n, b_n) \mathcal{N}(Z | \mu_n, \sigma_n^2) \ , \tag{6.31}$$

where $\mathrm{B}()$ is the Beta distribution. Now consider updating the posterior with the first measurement d_1. From Eq. (6.30)

$$q(Z, \pi | a_1, b_1, \mu_1, \sigma_1) = C \times p(d_1 | Z, \pi) q(Z, \pi | a_0, b_0, \mu_0, \sigma_0) \tag{6.32}$$

Where C is a normalisation constant such that $\int q(Z, \pi | a_1, b_1, \mu_1, \sigma_1) dZ d\pi = 1$. Eq. (6.32) is no longer of the form *Beta* $\times$ *Gaussian* violating Eq. (6.31). An approximation to Eq. (6.32) in the correct form can, however, be found using *moment matching*.

Here a new posterior $\tilde{q}()$ of the same form of $\mathrm{B} \times \mathcal{N}$ in Eq. (6.31) is used, whose first and second moments for Z and π are the same as those for the posterior in Eq. (6.32). The process of moment matching is particularly complex, involving solving a linear system of equations. The reader is therefore referred to the supplementary material of [32] for a full set of update equations.

When using depths for tracking, the depth used for the filter is simply the value of Z that maximises $q()$.

6.5 SLAM algorithm

An approach to direct structural refinement using least-squares has been described. To test the method, however, a full monocular SLAM method must be introduced. Details of the individual parts are discussed here.

As with other least-squares based methods [9, 18], the proposed algorithm does require an initialisation procedure to provide an initial map. This is discussed in Section 6.5.1. For real-time camera localisation, the direct method introduced in Chapter 5 is used. In Section 6.5.2, details of when and how the algorithm adds keyframes to the map are given.

6.5.1 Initialisation

The proposed method requires an initial map that is bootstrapped from a set of matches between two video frames.

Patch matching

Matching between frames begins with the detection of a set of corners in the first image frame I_{first}. The proposed method utilises Shi and Tomasi's "good features to track" [110] to provide a set of corners $\mathcal{V} = (\mathbf{v}_1, \mathbf{v}_2, ...)$. Each corner $\mathbf{v}_i$ is tracked using a direct formulation that estimates the translation $\mathbf{p}_{i1}$ that minimise:

$$\mathbf{p}_{i1} = \arg\max_{\mathbf{p}_{i1}} \sum_{\mathbf{u}_j \in \Omega_i} \left[I_{curr}(\mathbf{u}_j + \mathbf{p}_{i1}) - I_{first}(\mathbf{u}_j) \right]^2 , \tag{6.33}$$

where I_{curr} is the current video frame. Like tracking, this refinement is performed in a coarse to fine manner using an image pyramid. Once the process has converged, the match for the corner $\mathbf{v}_i$ is $(\mathbf{v}_i + \mathbf{p}_{i1})$. To provide a stronger set of matches reverse matching is performed. The premise of this process is that for a good match, the patch centred at $(\mathbf{v}_i + \mathbf{p}_{i1})$ in the current image I_{curr} should minimise the photometric error at $\mathbf{v}_i$ in the first image I_{first}. To test this, the translation $\mathbf{p}_{i2}$ that minimises

$$\mathbf{p}_{i2} = \arg\max_{\mathbf{p}_{i2}} \sum_{\mathbf{u}_j \in \Omega_i} \left[I_{curr}(\mathbf{u}_j + \mathbf{p}_{i1}) - I_{first}(\mathbf{u}_j + \mathbf{p}_{i2}) \right]^2 \tag{6.34}$$

is found. If the patch minimises photometric error at $\mathbf{v}_i$, the translation $\mathbf{p}_{i2}$ will have a norm of zero. Therefore good matches have a translation below the threshold T_{matching}:

$$\|\mathbf{p}_{i2}\| < T_{\text{matching}} \ , \tag{6.35}$$

where the closer T_{matching} is to zero, the stricter matching becomes.

Automatic initialisation

Many keyframe-based methods such as PTAM [9] require user intervention to select two initial keyframes used for the map. Relative pose estimation between them assumes a planar scene and estimates a homography which is ultimately decomposed into the relative pose.

Rather than require human intervention, we make use of the automatic initialisation method introduced by Mur-Artal *et al.* [18]. This method has two main benefits over PTAM's initialisation. Firstly, it is able to deal with both planar and general scenes, adaptively estimating the relative pose from either a homography or the fundamental matrix depending on how well each model fits the given matches. Secondly, it doesn't require human interaction as the method continuously monitors matches between the frames and only initialises once a good enough model has been found. The method is described here.

The method starts with a set matches between the first and second frame. From Section 6.5.1, we define the set of matches in the first frame I_{first} as:

$$\mathbf{x}_1 = \begin{bmatrix} \mathbf{v}_1 \\ \mathbf{v}_2 \\ \cdots \\ \mathbf{v}_N \end{bmatrix} \ , \tag{6.36}$$

and their corresponding matches in the second frame I_{curr} as

$$\mathbf{x}_2 = \begin{bmatrix} \mathbf{v}_1 + \mathbf{p}_{11} \\ \mathbf{v}_2 + \mathbf{p}_{21} \\ \cdots \\ \mathbf{v}_N + \mathbf{p}_{N1} \end{bmatrix} \ , \tag{6.37}$$

Given these matches, the automatic initialisation computes the homography `H` using [111] and the fundamental matrix `F` using [112]. A score is computed for each estimation is computed as follows:

$$S = \sum_i (\rho(d_{12}^2(\mathbf{x}_1^i, \mathbf{x}_2^i)) + \rho(d_{21}^2(\mathbf{x}_1^i, \mathbf{x}_2^i))) \,, \tag{6.38}$$

where d_{12} and d_{21} are symmetric transfer errors and

$$\rho(d^2) = \begin{cases} \Gamma - d^2 & \text{if } d^2 < T \\ 0 & \text{if } d^2 \geq T \end{cases} \,, \tag{6.39}$$

where T is an outlier rejection threshold (5.99 for homography, 3.84 for the fundamental matrix, which were determined using a χ^2 test at 95%). Γ is set equal to T for the homography estimation so that both models score equally for the same symmetric distance in their inlier region. Selection of the most reliable model is based on a on the robust statistic:

$$R_H = \frac{S_H}{S_H + S_F} \,, \tag{6.40}$$

where S_H and S_F are scores for the homography and fundamental matrix estimates respectively. If the initial scene is planar or nearly planer or the scene has low parallax, a homography best explains the camera movement. If the scene is not-planar and contains enough parallax, then the fundamental matrix should be used. The homography is selected for initialisation if $R_H \geq 0.45$, otherwise the fundamental matrix is used.

6.5.2 Keyframe addition

This section details how keyframes are added to the map and discusses the criteria used to determine whether the current frame should be used a keyframe or not.

The SLAM engine continuously monitors the distance between the current camera frame and the closest of the current keyframes in the map. If the following criteria are met, a new keyframe is initialised from the current frame and added to the map:

- The distance between the current frame and the closest keyframe exceeds a predefined threshold.

- The current error in the tracker is *below* a predefined threshold

Once a frame meets the criteria, it is turned into a keyframe and added to the map. Corners are then extracted from the image, and the depths for these corners are estimated by matching using the method in Section 6.5.1 and triangulation with other keyframes in the map. While other methods such as PTAM [9] search for matches using the closest other keyframe in the map, we search over a neighbourhood of 10 keyframes to utilise additional information which may not be present in a single keyframe.

6.6 Evaluation

6.6.1 Filtering versus bundle adjustment

Current state of the art sparsified direct methods utilise filtering for structural estimation [16, 34]. An alternative to this, in the form of a least-squares based direct bundle adjustment has been introduced. In this section, the accuracy of both bundle adjustment and filtering based methods are evaluated. The goal of here is to determine whether a large number of small-baseline measurements or a small number of large-baseline measurements are better in the context of a sparsified SLAM method. The TUM dataset is used for evaluation [104]. A number of test sequences are generated by selecting random 100-frame windows from random sequences and starting points. Each sequence is then processed for structural estimation using (i) a sparse, direct method first using a local bundle adjustment with 5 active frames, and (ii) filtering. The filtering method uses the same criteria for keyframes as the bundle adjustment method, performing update steps described in Section 6.4 to the closest keyframe after each pose refinement.

Both filtering and bundle adjustment share the same initial frame which is guaranteed to be a keyframe. To measure performance of the estimation, the estimated depths of corners in the first keyframe are evaluated. As the TUM dataset comes with a depth image for every frame, this is used as ground truth.

In terms of computational cost, filtering updates required 15ms per frame on the computer mentioned in Chapter 5. The bundle adjustment took an average of 100ms for all pyramid levels. Although bundle adjustment does require more time than filtering, we note that it is often performed in parallel to tracking, and therefore needn't run at frame-rate.

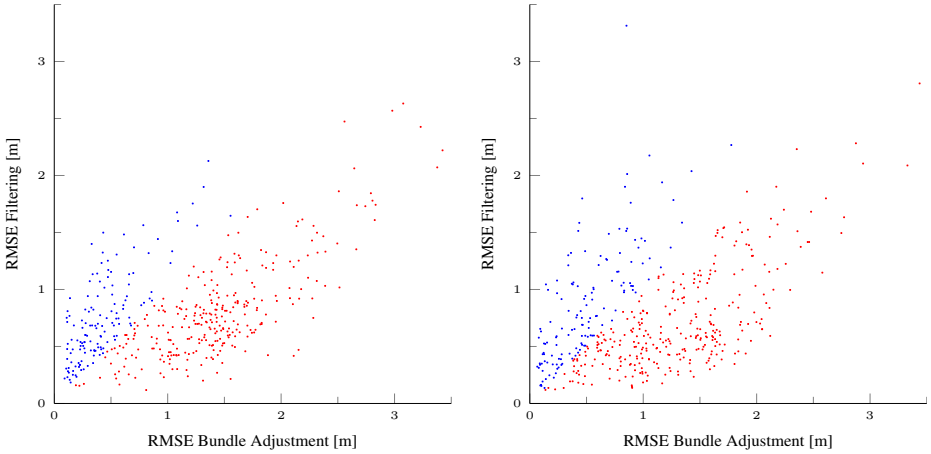
The evaluation uses a scale independent error in estimated depth. Given a set of corners $\mathbf{v}_1, \mathbf{v}_2 ..., \mathbf{v}_n$ with associated estimated depths $\{d_1, d_2, ...d_n\}$ in a frame.

$$E_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(sd_i - I_{\mathrm{depth}}(\mathbf{v}_i)\right)^2}, \tag{6.41}$$

where s is a scale factor that minimises E_{RMS}, and I_{depth} is the depth image that is associated with the frame.

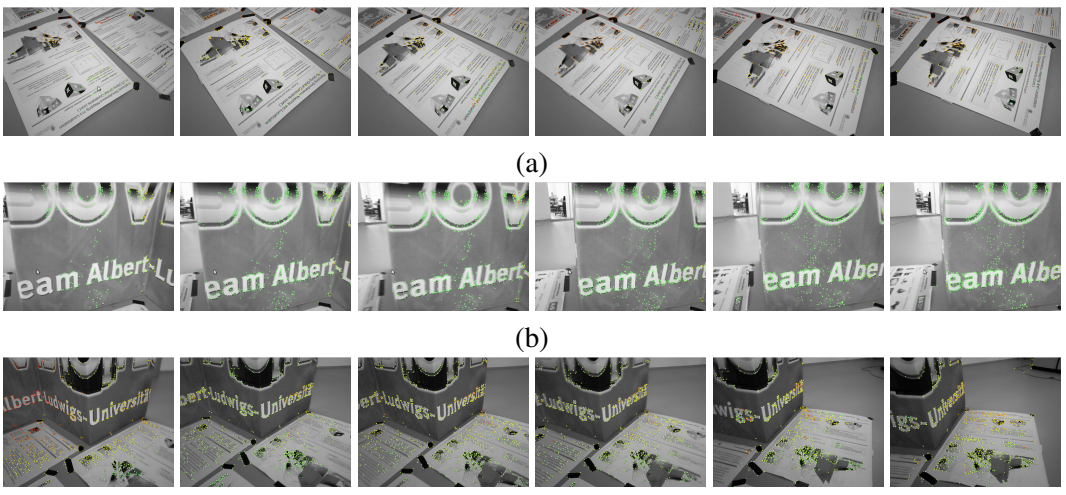
Fig. 6.3 shows the RMS error in estimated depth for a number of starting points in a number of sequences. For (a) ground-truth poses are provided while (b) requires the methods to estimate camera poses concurrently with depth. While results aren't particularly conclusive as to which of the methods is better, they are able to provide insight into which environments are better suited to each algorithm. We focus our attention on results *without* ground truth poses as this is a far more likely scenario in practice.

Fig. 6.4 shows three examples of sequences where bundle adjustment accuracy exceeds that of filtering. Bundle adjustment performs better for planar surfaces with minimal occlusion. One example of these sequences is shown in Fig. 6.5. Compare this to Fig. 6.6 which shows example sequences where filtering accuracy exceeds that of bundle adjustment. These sequences contain a large amount of occluding objects and changes in perspective. An example map from Fig. 6.6(c) is shown in Fig. 6.7 where occlusion from a plants leaves results in inaccurate depth estimates compared to the bottom half.



(a) (b)

Figure 6.3: RMS error in depth for first keyframe of a sequence using direct bundle adjustment vs using depth filtering. Results are shown using (a) ground-truth poses and (b) estimated poses. Red points are sequences where filtering performed better than bundle adjustment, while the converse is true for blue points.

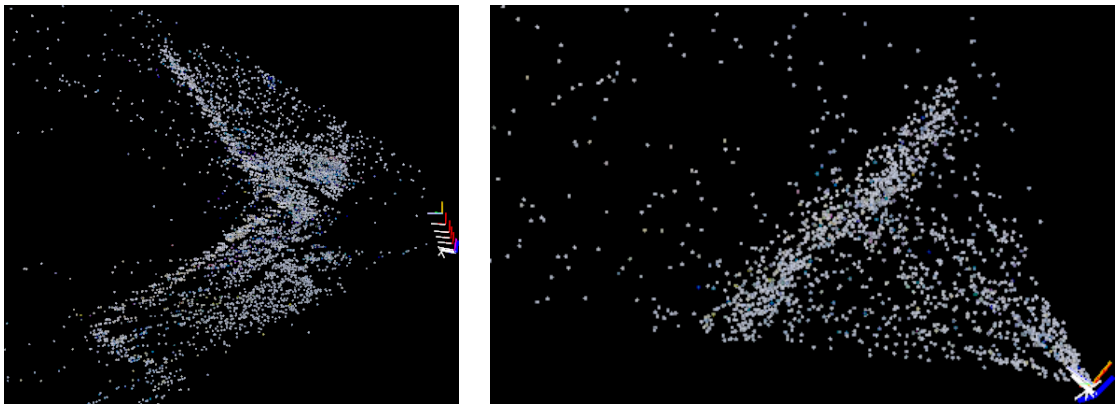


(a)

(b)

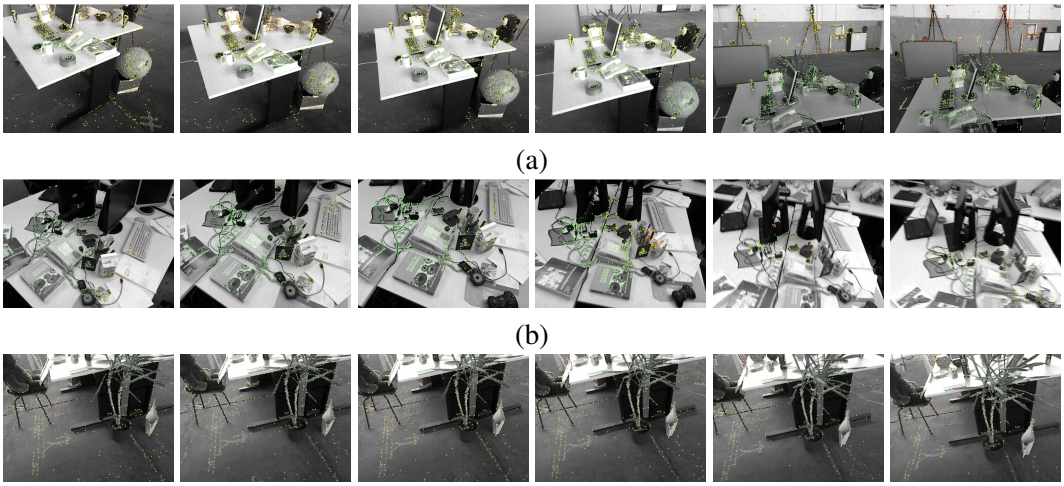
(c)

Figure 6.4: Example sequences where bundle adjustment accuracy exceeds filtering accuracy.



(a) (b)

Figure 6.5: Example estimated map from sequence (c) in Fig. 6.4. While (a) bundle adjustment is able to correctly estimate the structure of the map, (b) the filtering method is unable to do so.



(a)

(b)

(c)

Figure 6.6: Example sequences where filtering accuracy exceeds bundle adjustment accuracy.

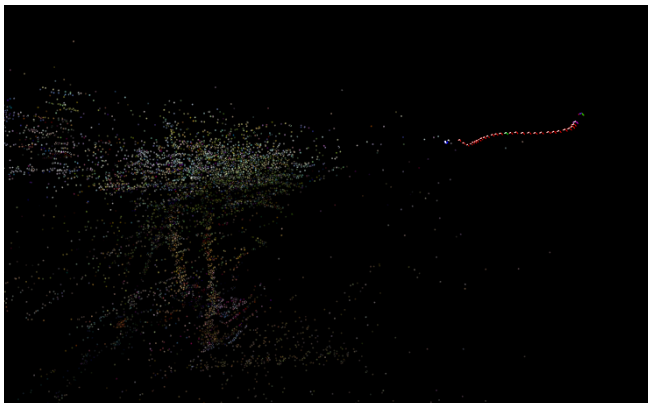
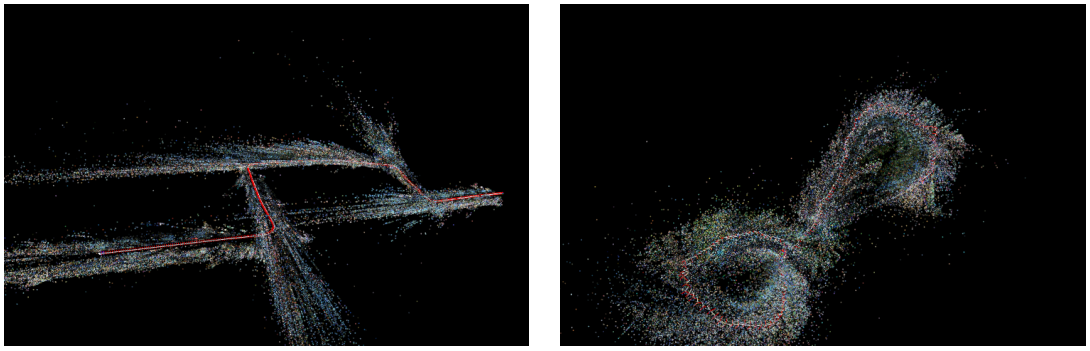


Figure 6.7: Example of inaccurate estimated map from sequence (c) in Fig. 6.6. While the bottom half of the plant is estimated correctly, occlusions due to its leaves result in inaccurate estimation at the top half.



(a) (b)

Figure 6.8: Resultant maps generated from running the (a) proposed method and (b) filtering on KITTI sequence 00.

6.6.2 Low frame-rate results

A particular benefit of methods that use least-squares over filtering is the ability to deal with low-frame rates. Filters generally require a large number of small-baseline measurements, while corners might only be observed a few times in low frame-rate video.

To demonstrate the ability of our proposed method to deal with low frame-rate video, the KITTI dataset is used [72, 50], which has a frame rate of 10Hz. Fig. 6.8(a) shows an example map that is generated from initial parts of sequence 00 of the KITTI odometry dataset using bundle adjustment. The estimated trajectory of the whole sequence vs ground truth is shown in Fig. 6.9. While the trajectory does indeed suffer from scale drift (although work from earlier chapters would indeed solve this), the trajectory is qualitatively accurate in translation and rotation. Fig. 6.8(b) shows an example map generated using a method using depth-filtering. As filters cannot converge properly with so few measurements, the estimated motion is chaotic in comparison.

As the use of low frame-rate video is for demonstrative purposes, we consider qualitative evaluation on the KITTI dataset unnecessary. Quantitative evaluation is considered in the next section.

6.6.3 Quantitative evaluation

To evaluate the performance of the proposed sparse direct SLAM system, the TUM dataset [104] is utilised.

Table 6.1 shows the absolute trajectory error for the proposed method and other monocular methods. Note that as all methods estimate the trajectory up to scale, results shown are for a

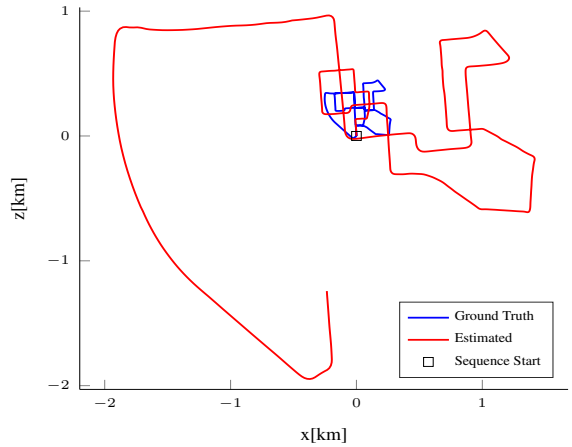


Figure 6.9: Estimated trajectory using proposed direct method vs ground truth for KITTI sequence 00. Despite scale drift, the estimated trajectory has minimal error when compared to ground truth.

sequence whose global scale is chosen to minimise error. Bundle adjustment is run locally using an active window of 5 frames. Results from individual sequences are discussed below.

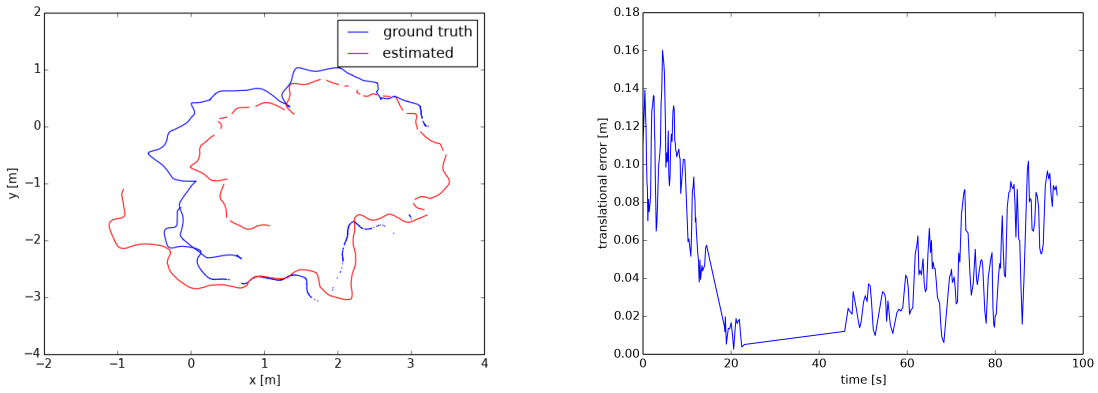
	Absolute Error [cm]			
Sequence	Ours	ORB-SLAM	PTAM	LSD-SLAM
freiburg2_desk	43.3	**0.88**	X	4.57
freiburg3_nostructure_texture_far	**2.0**	ambiguity	4.92	18.31
freiburg2_xyz	6.4	0.3	**0.2**	2.15
freiburg3_long_office	20.2(X)	**3.45**	X	38.53

Table 6.1: Absolute Keyframe RMSE for the proposed method versus other monocular SLAM methods. Results from other systems from [18]. "X" represents tracking failure on the sequence.

While the error for the *freiburg2_desk* sequence is comparatively larger than other methods, this is largely due to scale drift rather than an inaccurate construction. Fig. 6.10(a) shows the estimated trajectory for the sequence versus ground truth trajectory. Qualitatively the trajectories to appear to be fairly similar despite scale drift. This is confirmed by the relative per-keyframe error shown in Fig. 6.10(b), where error only exceeds 10cm near the beginning of the trajectory.

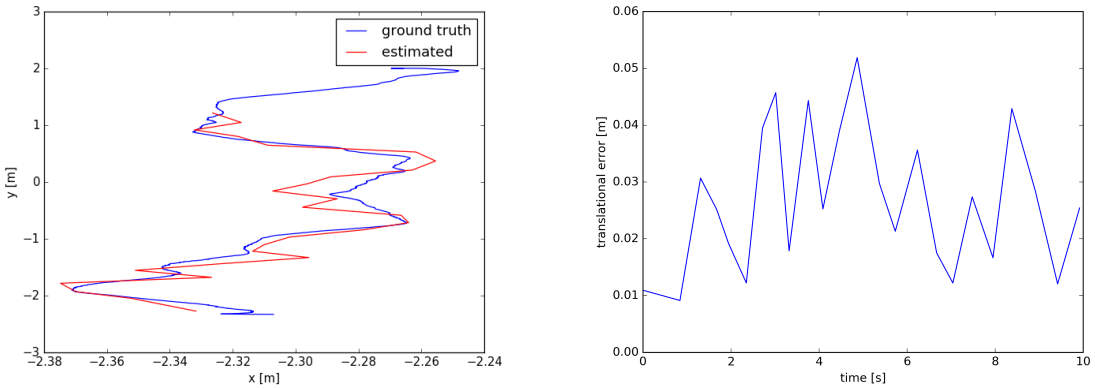
This sequence warrants discussion of loop closure. As there is a loop, methods with loop closure mechanisms such as ORB-SLAM and LSD-SLAM will undoubtedly have an advantage over a method that does not. Although loop-closure has not been implemented in this case, its addition through a double-window optimisation [55] is certainly possible.

Fig. 6.11 shows the estimated trajectory and relative translation error for the *nostructure_texture_far* sequence. As the proposed algorithm and ORB-SLAM share the same initialisation procedure, they suffer from an ambiguity where two relative pose estimates give rise to the same



(a) (b)

Figure 6.10: (a) Estimated trajectory and (b) relative translation error over time for freiburg2_desk sequence.



(a) (b)

Figure 6.11: (a) Estimated trajectory and (b) relative translation error over time for freiburg3_nostructure_texture_far sequence. Tracking eventually fails due to a lack of corners at the end of the trajectory.

matches between frames. For this reason, a homography estimation is used for initialisation (due to the scenes planar nature) and the results of the correct relative pose is used.

As tracking fails at the end of the sequence, the estimated trajectory is shorter than the ground-truth. The failure is shown in Fig. 6.12 where very few corners are detected at the end of the sequence. This sequence is an example case of where edge-based methods such LSD-SLAM[16] would be advantageous over the proposed corner-based method.

Fig. 6.13 shows the estimated trajectory for the *freiburg2_xyz* sequence. As this sequence does not contain loops, performance is comparable to LSD-SLAM, which is also a direct method.

The *freiburg3_nostructure_texture_far* sequence is particularly challenging for monocular meth-

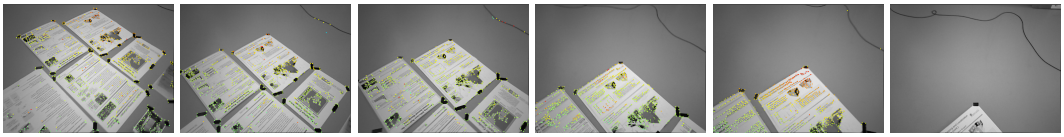
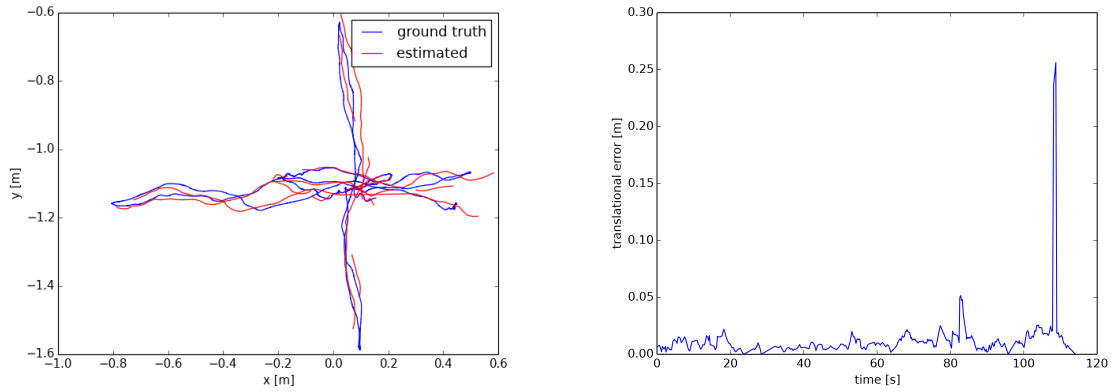


Figure 6.12: Lack of corners in freiburg3_nostructure_texture_far sequence leading to tracking failure.



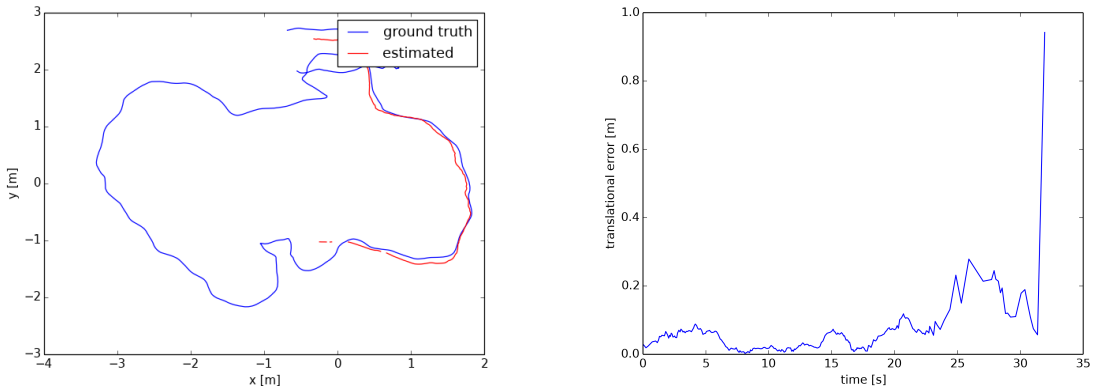
(a) (b)

Figure 6.13: (a) Estimated trajectory and (b) relative translation error over time for freiburg2_xyz sequence.

ods due to a number of pure rotations present in the sequence. Fig. 6.14 shows the estimated trajectory produced by the method up until the point of failure in a pure rotation. Frames from this point in the sequence are shown in Fig. 6.15. This highlights a particularly important benefit of filtering over bundle adjustment. Unlike bundle adjustment, which makes use of previous observations in earlier keyframes, filtering takes advantage of information in the future. For pure rotations, corners with high uncertainty can still be initialised and utilised for tracking, only to be updated with later measurements.

6.7 Conclusion

In this chapter a sparse, direct map-refinement method using least-squares rather than filtering has been presented. The method utilises a set of sparse corners in the image, and refines a set of keyframes and their associated inverse depths to minimise photometric error between them. Using an affine rather than a projective warp, the local nature of patches around corners is leveraged to reduce computational costs. For the purposes of comparison, a well-known depth-filtering method has been detailed as well.



(a) (b)

Figure 6.14: (a) Estimated trajectory and (b) relative translation error over time for freiburg3_long_office sequence. Estimated trajectory is shown in red and ground truth in blue. Tracking eventually fails due to a pure rotation in the sequence.

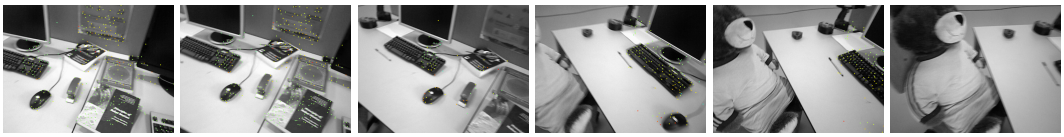


Figure 6.15: Example of a pure rotation in freiburg3_long_office, which leads to tracking failure

Evaluation shows that least-squares refinement is comparable with its filtering counterpart. A benefit of the method is its ability to deal with very low frame-rate video. While the least-squares method is able to accurately estimate structure in less crowded scenes, it does struggle when operating on sequences containing a lot of occlusion. Filtering methods generally converge on a good estimate quite quickly, and when a depth becomes occluded, successfully ignore subsequent measurements. Least-squares methods, on the other hand, do not have strong probabilistic priors and must initially treat photometric errors from all keyframes equally. This results in a struggle to differentiate whether a point in the map is occluded or not. As discussed in Section 6.2, multi-view stereo methods deal with this by either explicitly modelling visibility information or utilising a dense map in which visibility and occlusion is an intrinsic property that can be predicted from the map alone. A possible avenue of research is the modelling of visibility information for this work. This would naturally have to fit with computational constraints for real-time operation.

In terms of quantitative results, the proposed method has an accuracy that almost equals that of other sparsified direct methods on open-loop sequences. A significant feature that is lacking is loop-closure. In a sequence with a loop, scale drift that could otherwise be corrected using loop

closure added a large amount of error to the estimation. While other direct methods [16] use pose-graph optimisation alone, a joint bundle adjustment and pose graph optimisation as suggested by Strasdat *et al.* [55] could certainly be applied.

One downfall of bundle adjustment is inability to deal with pure rotations. Utilising filtering purely for initialisation is a possible solution to this. An interesting alternative is deferred triangulation [113], which, during a pure rotation, continues to track 2D corners and use them for rotation estimation. Only once the camera has experienced sufficient translation are corners triangulated.

Chapter 7

Conclusions and future work

This chapter summarises the contents of this thesis and its contributions to the field of monocular SLAM. Future research pertaining to each chapter is also suggested.

7.1 Summary of contributions

The aim of this thesis has been to present solutions to a common barrier to long-range single-camera SLAM; scale drift. Additionally it explored new solutions to a current trend of sparsification of direct methods.

After motivating for the research performed in this thesis in Chapter 1, three solutions to scale drift were explored. Each corrected scale drift by utilising prior knowledge about quantities that are affected by scale in monocular SLAM.

In Chapter 2 it was assumed that objects of a known class exist in the map. Objects were first localised by detecting them in multiple keyframes, and then optimised together with keyframes and landmarks in the map. If the size of objects is known prior to refinement, they are able to provide hints about the scale of the estimated map. Placing a prior on their size in a novel optimisation ensures that the rest of the map is consistent with their expected size. The proposed method was tested on a large outdoor dataset and obtained metric accuracy and corrected scale drift.

In Chapter 3 priors were placed on the depth of landmarks in video frames. A modified bundle adjustment was introduced that enforces depth priors on landmarks. This was first tested using LiDAR data and obtained metric accuracy without scale-drift on a large outdoor dataset. Next, an existing deep convolutional neural network that infers depth from a single image was used to

estimate an absolute depth map from each keyframe. The depth maps were used to provide depth priors to landmarks. Metric accuracy suffered slightly as the CNN used was unable to correctly estimate absolute depths, although relative depths were correctly estimated. As a result of correct relative depth estimation scale drift was successfully corrected.

Chapter 4 introduced a novel convolutional neural network that estimates the absolute speed of the camera from a pair of input images. Unique methods for augmenting the particular dataset used were also introduced. The speed estimated by the network was used to directly scale relative pose estimates from a monocular visual odometry algorithm. Although the method had difficulty estimating absolute speed, it accurately estimated relative speeds. The method was tested on a large outdoor dataset with minimal scale drift. Metric accuracy did suffer slightly due to incorrect absolute speed estimates.

The next two chapters explored direct solutions to monocular SLAM, with an emphasis on *sparse* corners rather than on entire images.

In Chapter 5 a novel sparsified direct tracker was introduced. Relying on corners in the image, the method leveraged the local nature of the projection function in the corner's neighbourhood. Projections were linearly approximated as an affine warp, which allowed for fast warping of a local image patch centred on the corner. This decreased computational cost. Additionally a naïve inverse additive formulation was introduced, however this was found to be slower and less accurate than its forward counterpart. The method was compared to a state of the art sparsified direct tracker that uses regions of high image-gradient. The proposed tracker was twice as fast as the state of the art while equal if not more accurate in terms of camera localisation. As computation time was reduced, tracking with multiple reference keyframes was explored. Evaluation showed that although multiple reference frames aid in tracking under heavy blur, they do sometimes decrease accuracy due to increased patch occlusions.

Chapter 6 continued with this research, introducing a method for sparse, direct structure-refinement using corners in the image. Rather than filtering using a large number of small-baseline measurements, the method attempts to use a smaller number of large baseline measurements in a novel bundle adjustment. Key to the bundle adjustment is the affine approximation introduced in the previous chapter, which allows a single corner to provide multiple pixel residuals to the bundle adjustment without requiring a separate projection for each of them. The method was compared with filtering in terms of accuracy of structural estimation. While the bundle adjustment was better

than filtering in some cases, the latter was shown to be better in dealing with occlusion in images. It was shown that a benefit of the proposed bundle adjustment was its ability to operate on video with a low frame-rate. Finally quantitative evaluation showed that while a SLAM method using the proposed bundle adjustment was able to obtain accuracy similar to current state of the art sparsified direct methods, occluded patches could lead to less accurate results.

7.2 Future work

As the bundle adjustment introduced in Chapter 2 operates in a local area around the current camera position, it is unable to propagate scale-corrections to other parts of the map. While a global bundle adjustment is infeasible due to computational restrictions, research has shown that local accuracy and global consistency may be obtained by performing a local bundle adjustment combined with a more global pose-graph optimisation [79]. Potential research could investigate adding pose-graph optimisation to the proposed bundle adjustment, while using similarity constraints between keyframes in the pose-graph [55] to allow scale to propagate along it. Additional object classes would increase the ability of the method to deal with more varied environments, although care has to be taken to properly sample object sizes. If metric accuracy isn't a concern, the size of objects may be estimated online in a method similar to [54]. Currently, object data association is performed independently from the bundle adjustment. Future work could investigate the benefits of utilising map information in object measurement data association and in moving object rejection.

In Chapter 3, the proposed network had difficulty estimating *absolute* depth, resulting in decreased metric accuracy. While this may be due to mismatched training and test domains, we argue that such a network might sacrifice accuracy for the sake of generalisation across the whole image. For sparse priors, this is not required. Future work might be to learn the depth for *specific* parts of the image that have interesting structure. These could additionally be constrained to patches with similar appearance that have a low variance in depth. Autoencoders [87] would allow for a low-level representation of patches to be learnt. This would allow for clustering of patches by their representation, and ultimately the rejection of clusters with high variation in their ground-truth depths.

The proposed deep convolutional network introduced in Chapter 4 is able to infer *relative*

speeds well, but is often unable to estimate the *absolute* speed of the camera accurately. The network was trained on a relatively specific domain using a small training set, and generalisation to other domains is therefore unlikely. Better generalisation and increased accuracy in absolute speed estimation can likely be obtained by using more training data. An advantage of this particular learning task is that, unlike many others, training examples can be obtained for "free". Combining a camera with an IMU would provide speed labels for as many training examples as desired. Additionally, recurrent neural networks could be investigated, as these architectures are more suited to the temporal nature of SLAM.

The methods introduced in Chapters 5 and 6 struggle to deal with occlusion of sparse patches. Rather than solely relying on a robust estimator to down-weight potentially occluded patches, future research could model visibility information during operation. This information could be used to remove occluded patches before they are included in refinement, rather than relying on the robust estimator alone. The SLAM algorithm using the proposed bundle adjustment often fails for sequences with pure rotations. Possible solutions to this include the use of filtering to initialise keyframes, or the use of deferred triangulation [113]. Here, 2D points are tracked when undergoing a pure rotation and triangulated later when a larger baseline becomes available. Additionally, the use of *double-window optimisation* [55] would allow for loop-closure and therefore increased accuracy.

Appendix A

Derivatives

A.1 Derivatives for rotations and euclidean transformations

A.1.1 Matrix exponential for rotations

The matrix exponential maps from the Lie-algebra $\boldsymbol{\phi} \in \mathfrak{so}(3) = [\phi_1, \phi_2, \phi_3]$ to Lie-group $\mathbf{SO}(3)$, which is the special orthogonal group of rotations in 3D. For $\mathbf{SO}(3)$ the exponential map is defined as:

$$\mathtt{R} = \exp(\boldsymbol{\phi}_\times) = I + \boldsymbol{\phi}_\times \frac{sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|} + \boldsymbol{\phi}_\times \frac{1 - cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2} \,, \tag{A.1}$$

where

$$\boldsymbol{\phi}_\times = \begin{bmatrix} 0 & -\phi_3 & \phi_2 \\ \phi_3 & 0 & -\phi_1 \\ -\phi_2 & \phi_1 & 0 \end{bmatrix} \,. \tag{A.2}$$

Which is the Rodriquez formula. If $\|\boldsymbol{\phi}\|$ is small, the resulting rotation matrix can be approximated as:

$$\mathtt{R} \approx I + \boldsymbol{\phi}_\times \tag{A.3}$$

The derivative of the rotation matrix with respect to the minimal parameters at $\phi = \mathbf{0}$, is simply:

$$\frac{\partial \mathtt{R}}{\partial \phi_1} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad \frac{\partial \mathtt{R}}{\partial \phi_2} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix} \quad \frac{\partial \mathtt{R}}{\partial \phi_3} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

A.1.2 Matrix exponential for euclidean transformations

Euclidean transformations in $\mathbf{SE}(3)$ can are parameterised by the Lie-algebra $\boldsymbol{\mu} = [\mathbf{t}, \boldsymbol{\phi}]^\top = [\mu_1, \mu_2, ..., \mu_6] \in \mathfrak{se}(3)$

$$\exp(\boldsymbol{\mu}) = \begin{bmatrix} \mathtt{R} & \mathbf{t} \\ 0^{1\times 3} & 1 \end{bmatrix}, \tag{A.4}$$

where $\mathbf{t}$ is the translation-component of the transformation, and $\mathtt{R}$ is a rotation matrix generated from $\boldsymbol{\phi}$. The derivatives of the transformation with respect to the elements of $\boldsymbol{\mu}$, $\frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_i}$ at $\boldsymbol{\mu} = \mathbf{0}$ are

$$\frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_1} = \mathtt{G}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_2} = \mathtt{G}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_3} = \mathtt{G}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_4} = \mathtt{G}_4 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_5} = \mathtt{G}_5 = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad \frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_6} = \mathtt{G}_6 = \begin{bmatrix} 0 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

A.2 Projection function

Given landmark in world coordinates $\mathbf{X}_{jw} \in \mathbb{R}^3$, and camera frame with pose $\mathtt{T}_i \in \mathbf{SE}(3)$, the landmark's position in the frame's coordinate system $\mathbf{X}_{ij}$ is:

$$\begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} = [X_1, X_2, X_3, 1]^\top = \mathtt{T}_j \begin{bmatrix} \mathbf{X}_{jw} \\ 1 \end{bmatrix} \tag{A.5}$$

The projection of the landmark in the camera plane is computed using the projection function proj:

$$\begin{bmatrix} \mathbf{x}_{ij} \\ 1 \end{bmatrix} = \mathrm{proj}(\mathbf{X}_{ij}) = \frac{1}{X_3} \begin{bmatrix} X_1 \\ X_2 \\ X_3 \end{bmatrix} \tag{A.6}$$

The derivative of the projected point with respect to $\mathbf{X}_{ij}$ is

$$\frac{d\mathbf{x}_{ij}}{d\mathbf{X}_{ij}} = \frac{1}{X_3} \begin{bmatrix} 1 & 0 & \frac{-X_1}{X_3} \\ 0 & 1 & \frac{-X_2}{X_3} \end{bmatrix} . \tag{A.7}$$

Points in the plane are mapped to their respective pixel locations using the intrinsic calibration matrix $\mathtt{K}$:

$$\mathtt{K} = \begin{bmatrix} f_u & 0 & c_u \\ 0 & f_v & c_v \\ 0 & 0 & 1 \end{bmatrix} , \tag{A.8}$$

where the pixels of the new point are computed as follows:

$$\begin{bmatrix} \mathbf{u}_{ij} \\ 1 \end{bmatrix} = \mathtt{K} \begin{bmatrix} \mathbf{x}_{ij} \\ 1 \end{bmatrix} \tag{A.9}$$

The derivative $\frac{d\mathbf{u}_{ij}}{d\mathbf{x}_{ij}}$ is simply the matrix:

$$\frac{d\mathbf{u}_{ij}}{d\mathbf{x}_{ij}} = \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \tag{A.10}$$

A.2.1 Derivative with respect to keyframe pose

For a small perturbation, the frame's pose $\mathtt{T}_i$ is updated as follows:

$$\mathtt{T}_j \leftarrow \exp(\boldsymbol{\mu})\mathtt{T}_j \ . \tag{A.11}$$

The landmark in the camera's frame therefore is:

$$\exp(\boldsymbol{\mu})\mathtt{T}_i \begin{bmatrix} \mathbf{X}_{jw} \\ 1 \end{bmatrix} = \exp(\boldsymbol{\mu}) \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} \ , \tag{A.12}$$

and the derivative of the landmark in the camera frame with respect to the perturbation is

$$\frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_i} \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} = \mathtt{G}_k \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} \ , \tag{A.13}$$

where $\mathtt{G}_k$ is the euclidean generator matrix defined in Section A.1.2. The matrix $\mathtt{J}_{\text{projcam}}$ is a 2×6 matrix and is defined as the derivative of the *pixel* projection $\mathtt{K}\ \text{proj}(\mathtt{T}_i, \exp(\boldsymbol{\mu})\mathbf{X}_{jw})$ respect to the perturbation $\boldsymbol{\mu}$:

$$\mathtt{J}_{\text{projcam}} = \frac{d\mathbf{u}_{ij}}{d\mu_k} = \frac{d\mathbf{u}_{ij}}{d\mathbf{x}_{ij}} \frac{1}{X_3} \begin{bmatrix} 1 & 0 & \frac{-X_1}{X_3} \\ 0 & 1 & \frac{-X_2}{X_3} \end{bmatrix} \mathtt{G}_k \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} \tag{A.14}$$

A.2.2 Derivative with respect to point

The derivative of the projection $\text{proj}(\mathtt{T}_i, \mathbf{X}_{jw})$ with respect to the point $\mathbf{X}_{jw}$ is

$$\frac{d\text{proj}(\mathtt{T}_i, \mathbf{X}_{jw})}{d\mathbf{X}_{jw}} = \frac{d\text{proj}(\mathtt{R}_i\mathbf{X}_{jw} + \mathbf{t}_i)}{d\mathbf{X}_{jw}} = \mathtt{R}_i \tag{A.15}$$

The matrix $\mathtt{J}_{\text{projpoint}}$ is a 2×3 matrix and is defined as the derivative of the *pixel* projection $\mathtt{K}\ \text{proj}(\mathtt{T}_i, \mathbf{X}_{jw})$ with respect to the point $\mathbf{X}_{jw}$:

$$\mathtt{J}_{\text{projpoint}} = \frac{d\mathbf{u}_{ij}}{d\mathbf{X}_{jw}} = \frac{d\mathbf{u}_{ij}}{d\mathbf{x}_{ij}} \frac{1}{X_3} \begin{bmatrix} 1 & 0 & \frac{-X_1}{X_3} \\ 0 & 1 & \frac{-X_2}{X_3} \end{bmatrix} \mathtt{R}_i \tag{A.16}$$

A.3 3D point of back-projected inverse depth

For a corner $\mathbf{v}_j$ with inverse depth ρ_j in a keyframe with pose $\mathtt{T}_i$, the resultant point in the camera's coordinate system is:

$$\mathbf{X}_{ij} = \mathrm{proj}^{-1}(\mathbf{x}_j, \rho_j) = \begin{bmatrix} x_{j1}/\rho_j \\ x_{j2}/\rho_j \\ 1/\rho_j \end{bmatrix} , \tag{A.17}$$

where proj^{-1} is the back-projection function and $\mathbf{x}_j = [x_{j1}, x_{j2}]$ are the coordinates of the corner on the camera's image plane:

$$\begin{bmatrix} \mathbf{x}_j \\ 1 \end{bmatrix} = \mathtt{K}^{-1} \begin{bmatrix} \mathbf{v}_j \\ 1 \end{bmatrix} . \tag{A.18}$$

The position of the point in the world frame is

$$\begin{bmatrix} \mathbf{X}_{jw} \\ 1 \end{bmatrix} = \mathtt{T}_i^{-1} \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} . \tag{A.19}$$

A.3.1 Derivative with respect to inverse depth

The derivative of point in the cameras coordinate system $\mathbf{X}_{ij}$ with respect to the inverse depth ρ is

$$\frac{\partial \mathbf{X}_{ij}}{\partial \rho} = \begin{bmatrix} -x_1/\rho^2 \\ -x_2/\rho^2 \\ -1/\rho^2 \end{bmatrix} . \tag{A.20}$$

We define the matrix $\mathtt{J}_{\mathrm{IDDepth}}$ as the derivative of the point in world coordinates $\mathbf{X}_{jw}$ with respect to inverse depth ρ_j

$$\mathtt{J}_{\mathrm{IDDepth}} = \frac{d\mathbf{X}_{jw}}{d\rho_j} = \mathtt{R}_i^{-1} \begin{bmatrix} -x_1/\rho_j^2 \\ -x_2/\rho_j^2 \\ -1/\rho_j^2 \end{bmatrix} , \tag{A.21}$$

where $\mathtt{R}_i$ is the rotation matrix of $\mathtt{T}_i$

A.3.2 Derivative with respect to source keyframe pose

For a small perturbation $\boldsymbol{\mu}$ to the source keyframe's pose $\mathtt{T}_i$, the new point in world coordinates is:

$$\begin{bmatrix} \mathbf{X}_{jw} \\ 1 \end{bmatrix} = (\exp(\boldsymbol{\mu})\mathtt{T}_i)^{-1} \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} = \mathtt{T}^{-1}\exp(\boldsymbol{\mu})^{-1} \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} \tag{A.22}$$

$$= \mathtt{T}^{-1}\exp(-\boldsymbol{\mu}) \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} = -\mathtt{T}^{-1}\exp(\boldsymbol{\mu}) \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} \tag{A.23}$$

The derivative of the point in the world coordinate system $\mathbf{X}_{jw}$ with respect to an element μ_k of $\boldsymbol{\mu}$ is

$$\frac{d\mathbf{X}_{jw}}{d\mu_k} = -\mathtt{R}_i^{-1}\mathtt{G}_k \begin{bmatrix} \mathbf{X}_{ij} \\ 1 \end{bmatrix} , \tag{A.24}$$

where $\mathtt{G}_k$ is the euclidean generator matrix for μ_k defined in Section A.1.2. The matrix $\mathtt{J}_{\mathrm{IDsource}}$ is defined as:

$$\mathtt{J}_{\mathrm{IDsource}} = \begin{bmatrix} \frac{d\mathbf{X}_{jw}}{d\mu_1} & \frac{d\mathbf{X}_{jw}}{d\mu_2} & \frac{d\mathbf{X}_{jw}}{d\mu_3} & \frac{d\mathbf{X}_{jw}}{d\mu_4} & \frac{d\mathbf{X}_{jw}}{d\mu_5} & \frac{d\mathbf{X}_{jw}}{d\mu_6} \end{bmatrix} \tag{A.25}$$

Appendix B

Image parameterisation

The direct methods discussed in this thesis minimise photometric error. This requires knowledge of how images are parameterised, how their gradients are computed, and how sub-pixel interpolation is used to ensure images are smooth functions for continuous optimisation.

B.1 Image parameterisation

Images are considered as a discrete multivariate function that takes image coordinates $\mathbf{u} = [u_1, u_2], u \in \mathbb{Z}^+$:

$$I(\mathbf{u}) : \mathbb{Z}^2 \rightarrow \mathbb{R} \, . \tag{B.1}$$

The function $I()$ is generally considered non-linear as the values it returns are essentially unrelated to its inputs. Optimisation often requires a gradient for the function which is approximated from second order methods:

$$\nabla I = \left[\frac{\partial I}{\partial u_1}, \frac{\partial I}{\partial \, , u_2} \right] \, , \tag{B.2}$$

where

$$\begin{aligned} \left. \frac{\partial I}{\partial u_1} \right|_{\mathbf{u}} &\approx \frac{1}{2} \left[I(\mathbf{u} + [1,0]^\top) - I(\mathbf{u} - [1,0]^\top) \right] \\ \left. \frac{\partial I}{\partial u_2} \right|_{\mathbf{u}} &\approx \frac{1}{2} \left[I(\mathbf{u} + [0,1]^\top) - I(\mathbf{u} - [0,1]^\top) \right] \end{aligned} \, . \tag{B.3}$$

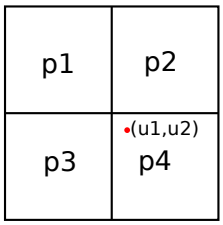


Figure B.1: Sub-pixel interpolation

B.2 Sub-pixel interpolation

Fig. B.1 shows a sample of how sub-pixel interpolation is performed. For the pixel $\mathbf{u} = [u_1, u_2], u \in \mathbb{R}^+$

$$I(\mathbf{u}) = p_1 f_1 + p_2 f_2 + p_3 f_3 + p_4 f_4 \tag{B.4}$$

where

$$\begin{aligned} f_1 &= (1 - d_1)(1 - d_2) \\ f_2 &= (d_1)(1 - d_2) \\ f_3 &= (1 - d_1)d_2 \\ f_4 &= d_1 d_2 \end{aligned} \tag{B.5}$$

and

$$\begin{aligned} d_1 &= u_1 - \lfloor u_1 \rfloor \\ d_2 &= u_2 - \lfloor u_2 \rfloor \end{aligned} \tag{B.6}$$

Bibliography

[1] H. C. Longuet-Higgins., "A computer algorithm for reconstructing a scene from two projections." *Nature*, vol. 293, pp. 133–135, 1981.

[2] M. Pollefeys, L. V. Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch, "Visual modeling with a hand-held camera," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 207–232, 2004.

[3] J. J. Leonard, H. F. Durrant-Whyte, and I. J. Cox, "Dynamic map building for an autonomous mobile robot," *International Journal of Robotics Research*, vol. 11, no. 8, pp. 286–298, 1992.

[4] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc 9th IEEE Int Conf on Computer Vision*, 2003, pp. II: 1403–1410.

[5] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 6, pp. 1052–1067, 2007.

[6] R. C. Smith and P. Cheeseman, "On the representation and estimation of spatial uncertainty," *International Journal of Robotics Research*, vol. 5, no. 4, pp. 56–68, 1986.

[7] A. J. Davison and D. W. Murray, "Mobile robot localisation using active vision," in *Proc 5th European Conf on Computer Vision, Freiburg, Germany, May*, ser. LNCS. Springer-Verlag, 1998, pp. 809–825.

[8] ——, "Sequential localisation and map-building using active vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 865–880, Jul. 2002.

[9] G. Klein and D. W. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc 6th IEEE/ACM Int Symp on Mixed and Augmented Reality*, 2007, pp. 225–234.

[10] M. Okutomi and T. Kanade, "A multiple-baseline stereo," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 15, no. 4, pp. 353–363, Apr. 1993.

[11] R. A. Newcombe and A. J. Davison, "Live dense reconstruction with a single moving camera," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2010, pp. 1498–1505.

[12] J. Stühmer, S. Gumhold, and D. Cremers, "Real-time dense geometry from a handheld camera," in *Joint Pattern Recognition Symposium*. Springer, 2010, pp. 11–20.

[13] V. Pradeep, C. Rhemann, S. Izadi, C. Zach, M. Bleyer, and S. Bathiche, "Monofusion: Real-time 3d reconstruction of small scenes with a single web camera," in *2013 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 83–88.

[14] R. A. Newcombe, S. Lovegrove, J. Steven, and A. J. Davison, "DTAM: Dense tracking and mapping in real-time," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2011, pp. 2320–2327.

[15] P. Ondrúška, P. Kohli, and S. Izadi, "Mobilefusion: Real-time volumetric surface reconstruction and dense tracking on mobile phones," *IEEE transactions on visualization and computer graphics*, vol. 21, no. 11, pp. 1251–1258, 2015.

[16] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 834–849.

[17] R. Wiseman, "Assumptions," http://www.youtube.com/watch?v=zNbF006Y5x4, 2012, accessed: 27-07-2013.

[18] R. Mur-Artal, J. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[19] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment-a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

[20] H. Strasdat, "Local accuracy and global consistency for efficient visual SLAM," Ph.D. dissertation, Imperial College London, 2012.

[21] C. Mei, G. Sibley, M. Cummins, P. Newman, and I. Reid, "RSLAM: a system for large-scale mapping in constant time using stereo," *International Journal of Computer Vision*, vol. 94, no. 2, pp. 198–214, 2011.

[22] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *IEEE ISMAR*. IEEE, 2011.

[23] O. Kahler, V. A. Prisacariu, C. Y. Ren, X. Sun, P. H. S. Torr, and D. W. Murray, "Very High Frame Rate Volumetric Integration of Depth Images on Mobile Device," *IEEE Transactions on Visualization and Computer Graphics (Proceedings International Symposium on Mixed and Augmented Reality 2015*, vol. 22, no. 11, 2015.

[24] J. Zhang and S. Singh, "Visual-LiDAR odometry and mapping: Low-drift, robust, and fast," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[25] V. A. Prisacariu, O. Kähler, M. Cheng, C. Y. Ren, J. P. C. Valentin, P. H. S. Torr, I. D. Reid, and D. W. Murray, "A framework for the volumetric integration of depth images," *CoRR*, vol. abs/1410.0925, 2014. [Online]. Available: http://arxiv.org/abs/1410.0925

[26] O. Kahler, V. Prisacariu, C. Ren, X. Sun, P. Torr, and D. Murray, "Very high frame rate volumetric integration of depth images on mobile devices," *Visualization and Computer Graphics, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2015.

[27] B. Julesz, *Foundations of Cyclopean Perception*. University of Chicago Press, 1971.

[28] J. M. Loomis, "Looking down is looking up," *Nature*, vol. 414, no. 6860, pp. 155–156, 2001.

[29] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[31] R. O. Castle, G. Klein, and D. W. Murray, "Video-rate localization in multiple maps for wearable augmented reality," in *Proc 12th IEEE Int Symp on Wearable Computers*, 2008.

[32] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, no. 7, pp. 434–441, 2011.

[33] M. Pizzoli, C. Forster, and D. Scaramuzza, "REMODE: Probabilistic, monocular dense reconstruction in real time," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 2609–2616.

[34] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 15–22.

[35] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *IEEE International Conference on Computer Vision (ICCV)*. IEEE, 2013, pp. 1449–1456.

[36] J. Engel, V. Kolton, and D. Cremers, "Direct sparse odometry," in *arXiv:1607.02565*, 2016.

[37] H. Strasdat, J. Montiel, and A. J. Davison, "Real-time monocular SLAM: Why filter?" in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2010, pp. 2657–2664.

[38] D. P. Frost, O. Kahler, and D. W. Murray, "Object-aware bundle adjustment for correcting monocular scale drift," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2016, pp. 4770–4776.

[39] J.-S.Gutmann and K. Konolige, "Incremental mapping of large cyclic environments," in *Proc IEEE Int Symp on Computational Intelligence in Robotics and Automation*, 1999, pp. 318–325.

[40] S. Se, D. G. Lowe, and J. J. Little, "Local and global localization for mobile robots using visual landmarks," *Proc IEEE/RSJ Conf on Intelligent Robots and Systems, Lausanne, Switzerland, Oct 2-4, 2002*, pp. 414–420, 2002.

[41] M. Cummins and P. Newman, "FAB-MAP: Probabilistic localization and mapping in the space of appearance," *International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[42] O. Kähler, V. A. Prisacariu, and D. W. Murray, "Real-time large-scale dense 3d reconstruction with loop closure," in *European Conference on Computer Vision*. Springer, 2016, pp. 500–516.

[43] R. O. Castle, G. Klein, and D. W. Murray, "Combining monoSLAM with object recognition for scene augmentation using a wearable camera," *Image and Vision Computing*, vol. 28, no. 12, pp. 1548–1556, 2010.

[44] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*. IEEE, 2015, pp. 1935–1942.

[45] S.-H. Jung and C. J. Taylor, "Camera trajectory estimation using inertial sensor measurements and structure from motion results," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2. IEEE, 2001, pp. II–732.

[46] G. Nützi, S. Weiss, D. Scaramuzza, and R. Siegwart, "Fusion of IMU and vision for absolute scale estimation in monocular SLAM," *Journal of Intelligent and Robotic Systems*, vol. 61, no. 1-4, pp. 287–299, 2011.

[47] C. Hide, T. Botterill, and M. Andreotti, "Vision-aided IMU for handheld pedestrian navigation," in *Proceedings of the institute of navigation GNSS 2010 Conference, Portland, Oregon*, 2010.

[48] M. Achtelik, M. Achtelik, S. Weiss, and R. Siegwart, "Onboard IMU and monocular vision based control for mavs in unknown in-and outdoor environments," in *IEEE international conference on Robotics and automation (ICRA)*. IEEE, 2011, pp. 3056–3063.

[49] A. Geiger, J. Ziegler, and C. Stiller, "Stereoscan: Dense 3D reconstruction in real-time," in *Proc Intelligent Vehicles Symposium IV*, 2011, pp. 963–968.

[50] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *Proc Intelligent Vehicles Symposium III*, 2010, pp. 486–492.

[51] D. Scaramuzza, F. Fraundorfer, M. Pollefeys, and R. Siegwart, "Absolute scale in structure from motion from a single vehicle mounted camera by exploiting nonholonomic constraints," in *Proc 12th IEEE Int Conf on Computer Vision*. IEEE, 2009, pp. 1413–1419.

[52] S. Song, M. Chandraker, and C. C. Guest, "Parallel, real-time monocular visual odometry," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 4698–4705.

[53] S. Song and M. Chandraker, "Robust scale estimation in real-time monocular SFM for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2014, pp. 1566–1573.

[54] T. Botterill, S. Mills, and R. D. Green, "Correcting scale drift by object recognition in single-camera SLAM," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1767–1780, 2013.

[55] H. Strasdat, J. M. Montiel, and A. J. Davison, "Drift aware large scale monocular SLAM," in *Proc Robotics: Science and Systems Conference*, 2010.

[56] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[57] R. O. Castle and D. W. Murray, "Keyframe-based recognition and localization during video-rate parallel tracking and mapping," *Image and Vision Computing*, vol. 29, no. 8, pp. 524–532, 2011.

[58] J. Civera, D. Gálvez-López, L. Riazuelo, , J. D. Tardós, and J. M. M. Montiel, "Towards semantic SLAM using a monocular camera," in *Proc IEEE/RSJ Conf on Intelligent Robots and Systems, San Francisco CA, USA, Sep 25-30, 2011*, 2011, pp. 1277–1284.

[59] S. Y. Bao and S. Savarese, "Semantic structure from motion," in *Proc 24th IEEE Conf on Computer Vision and Pattern Recognition*, 2011, pp. 2025–2032.

[60] D. Gálvez-López, M. Salas, and J. M. M. Monteil, "Real-time monocular object SLAM," `arXiv:1504.02398 [cs.CV]`, 2015.

[61] A. Dame, V. A. Prisacariu, C. Y. Ren, and I. Reid, "Dense reconstruction using 3D object shape priors," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2013, pp. 1288–1295.

[62] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.

[63] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1. IEEE, 2005, pp. 886–893.

[64] S. Y. Bao, M. Bagra, and S. Savarese, "Semantic structure from motion with object and point interactions," in *Proc Workshops, 13th IEEE Conf on Computer Vision*, 2011, pp. 982–989.

[65] S. Y. Bao, M. Bagra, Y.-W. Chao, and S. Savarese, "Semantic structure from motion with points, regions, and objects," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2012, pp. 2703–2710.

[66] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *International Conference on Computer Vision (ICCV)*, 2013.

[67] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D traffic scene understanding from movable platforms," *Pattern Analysis and Machine Intelligence (PAMI)*, 2014.

[68] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[69] P. J. Huber, *Robust statistics*. Springer, 2011.

[70] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc 9th European Conf on Computer Vision*, 2006.

[71] C. Kerl, J. Sturm, and D. Cremers, "Dense visual SLAM for RGB-D cameras," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2013, pp. 2100–2106.

[72] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[73] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *Proc 14th IEEE Int Conf on Computer Vision*, 2013, pp. 3056–3063.

[74] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D traffic scene understanding from movable platforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 1012–1025, 2014.

[75] H. Strasdat, J. Montiel, and A. J. Davison, "Scale drift-aware large scale monocular SLAM." in *Robotics: Science and Systems*, vol. 2, no. 3, 2010, p. 5.

[76] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI vision benchmark suite," in *Proc 25th IEEE Conf on Computer Vision and Pattern Recognition*, 2012, pp. 3354–3361.

[77] R. Kümmerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, "On measuring the accuracy of slam algorithms," *Autonomous Robots*, vol. 27, no. 4, pp. 387–407, 2009.

[78] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International journal of computer vision*, vol. 56, no. 3, pp. 221–255, 2004.

[79] H. Strasdat, A. J. Davison, J. M. M. Montiel, and K. Konolige, "Double window optimisation for constant time visual SLAM," in *Proc 13th IEEE Int Conf on Computer Vision*. IEEE, 2011, pp. 2352–2359.

[80] J. Michels, A. Saxena, and A. Y. Ng, "High speed obstacle avoidance using monocular vision and reinforcement learning," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 593–600.

[81] A. Saxena, S. H. Chung, and A. Y. Ng, "Learning depth from single monocular images," in *Advances in Neural Information Processing Systems*, 2005, pp. 1161–1168.

[82] A. Saxena, M. Sun, and A. Y. Ng, "Make3d: Learning 3D scene structure from a single still image," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 5, pp. 824–840, 2009.

[83] D. Eigen, C. Puhrsch, and R. Fergus, "Depth map prediction from a single image using a multi-scale deep network," in *Advances in neural information processing systems*, 2014, pp. 2366–2374.

[84] F. Liu, C. Shen, and G. Lin, "Deep convolutional neural fields for depth estimation from a single image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5162–5170.

[85] F. Liu, C. Shen, G. Lin, and I. Reid, "Learning depth from single monocular images using deep convolutional neural fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 10, pp. 2024–2039, 2016.

[86] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus, "Indoor segmentation and support inference from RGBD images," in *European Conference on Computer Vision*. Springer, 2012, pp. 746–760.

[87] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[88] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *IEEE International Conference on Computer Vision*. IEEE, 2003, pp. 1470–1477.

[89] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[90] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[91] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting." *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[92] S. Zagoruyko and N. Komodakis, "Learning to compare image patches via convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4353–4361.

[93] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence,*, vol. 37, no. 9, pp. 1904–1916, 2015.

[94] 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," *arXiv preprint arXiv:1504.06852*, 2015.

[95] K. Konda and R. Memisevic, "Learning visual odometry with a convolutional network," in *International Conference on Computer Vision Theory and Applications*. Citeseer, 2015.

[96] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. [Online]. Available: http://arxiv.org/abs/1605.02688

[97] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[98] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European Conference on Computer Vision*. Springer, 2014, pp. 818–833.

[99] S. Lovegrove, A. J. Davison, and J. Ibanez-Guzmán, "Accurate visual odometry from a rear parking camera," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*. IEEE, 2011, pp. 788–793.

[100] T. Sato, M. Kanbara, N. Yokoya, and H. Takemura, "Dense 3-D reconstruction of an outdoor scene by hundreds-baseline stereo using a hand-held video camera," *International Journal of Computer Vision*, vol. 47, no. 1-3, pp. 119–129, 2002.

[101] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the 7th International Joint Conference on Artificial Intelligence*, 1981, pp. 674–679.

[102] G. D. Hager and P. N. Belhumeur, "Efficient region tracking with parametric models of geometry and illumination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 10, pp. 1025–1039, 1998.

[103] P. Huber, *Robust statistics*. John Wiley and Sons, 1985.

[104] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *Proc. of the International Conference on Intelligent Robot Systems (IROS)*, Oct. 2012.

[105] Y. Furukawa and J. Ponce, "Accurate, dense, and robust multiview stereopsis," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 8, pp. 1362–1376, 2010.

[106] ——, "Accurate camera calibration from multi-view stereo and bundle adjustment," *International Journal of Computer Vision*, vol. 84, no. 3, pp. 257–268, 2009.

[107] A. Delaunoy and M. Pollefeys, "Photometric bundle adjustment for dense multi-view 3D modeling," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2014, pp. 1486–1493.

[108] L. Matthies, R. Szeliski, and T. Kanade, "Incremental estimation of dense depth maps from image sequences," in *Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1988, pp. 366–374.

[109] J. Civera, A. J. Davison, and J. M. Montiel, "Inverse depth parametrization for monocular SLAM," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, 2008.

[110] J. Shi and C. Tomasi, "Good features to track," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 1994, pp. 593–600.

[111] O. D. Faugeras and F. Lustman, "Motion and structure from motion in a piecewise planar environment," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, no. 3, pp. 485–508, 1988.

[112] W. Tan, H. Liu, Z. Dong, G. Zhang, and H. Bao, "Robust monocular SLAM in dynamic environments," in *IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, 2013, pp. 209–218.

[113] C. D. Herrera, K. Kim, J. Kannala, K. Pulli, and J. Heikkilä, "DT-SLAM: Deferred triangulation for robust SLAM," in *2014 2nd International Conference on 3D Vision*, vol. 1. IEEE, 2014, pp. 609–616.