Model-Based Segmentation Methods for Analysis of 2D and 3D Ultrasound Images and Sequences

Richard Stebbing

St Peter's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2014

Acknowledgements

As I write this final section of my thesis I am acutely aware of how many things had to fall into place for me to be in Oxford and for this to come together. Frankly, it would never have happened without the academic and personal support of so many people. I would like to thank:

The Rhodes Trust, for making it possible for me to study and live in Oxford. The last four years have been an incredible personal, intellectual, and academic learning experience for me. I will be forever grateful for this opportunity and the chance you have taken on me.

The North Harbour Club, for their support and funding of my fourth year.

St Peter's College, for taking me on as Assistant Junior Dean.

My supervisor, Alison Noble, who has been incredibly encouraging and supportive of my foray into ultrasound image analysis and academia.

Andrew Fitzgibbon, who took me on as an intern twice at Microsoft Research and whose guidance fundamentally changed my approach to computer vision.

My partners in crime in the Biomedical Image Analysis Laboratory.

My friends in Oxford from the Rhodes community, the powerlifting club, Zappi's, Vinnie's, St John's, St Peter's, and beyond. You all have been my family away from home, encouraging me at the best of times and lifting me up at the toughest.

Finally, my family — James, Carolyn, Robyn, and Vaughan — who have cheered me on from the other side of the world. This is for you.

Abstract

This thesis describes extensions to 2D and 3D model-based segmentation algorithms for the analysis of ultrasound images and sequences.

Starting from a common 2D+t "track-to-last" algorithm, it is shown that the typical method of searching for boundary candidates perpendicular to the model contour is unnecessary if, for each boundary candidate, its corresponding position on the model contour is optimised *jointly* with the model contour geometry. With this observation, two 2D+t segmentation algorithms, which accurately recover boundary displacements and are capable of segmenting arbitrarily long sequences, are formulated and validated.

Generalising to 3D, subdivision surfaces are shown to be natural choices for continuous model surfaces, and the algorithms necessary for joint optimisation of the correspondences and model surface geometry are described. Three applications of 3D model-based segmentation for ultrasound image analysis are subsequently presented and assessed: skull segmentation for fetal brain image analysis; face segmentation for shape analysis, and single-frame left ventricle (LV) segmentation from echocardiography images for volume measurement. A framework to perform model-based segmentation of *multiple* 3D sequences — while jointly optimising an underlying linear basis shape model — is subsequently presented for the challenging application of right ventricle (RV) segmentation from 3D+t echocardiography sequences. Finally, an algorithm to automatically select boundary candidates *independent* of a model surface estimate is described and presented for the task of LV segmentation.

Although motivated by challenges in ultrasound image analysis, the conceptual contributions of this thesis are general and applicable to model-based segmentation problems in many domains. Moreover, the components are modular, enabling straightforward construction of application-specific formulations for new clinical problems as they arise in the future.

Contents

Chapter 1

Introduction

Diagnostic sonography, otherwise known as *ultrasound*, is an imaging modality that uses high frequency sound waves to form 2D and 3D images of an anatomical structure [69]. The advantages of ultrasound imaging are that: (a) 2D and 3D ultrasound imaging systems are portable, (b) it does not expose the subject to ionizing radiation, (c) acquisition of image sequences can be performed in real-time, and (d) the speckle pattern, which is often considered an undesirable imaging artefact, moves with the underlying tissue [43]. Disadvantages of ultrasound imaging include: (a) it cannot image anatomical structures occluded by bone, which limits the field of view in some applications, and (b) interference from adipose tissue causes non-uniform signal attenuation and reduces the contrast of the final image.

Clinical applications of diagnostic ultrasound pertinent to this thesis are:

1. **Cardiology** *Echocardiography* is the use of ultrasound for the assessment of heart structure and function [86]. An *echocardiogram* is a 2D or 3D ultrasound image of the heart (Figure 1.1) and is primarily used for qualitative assessment of heart structure and function. For quantitative assessment, the normalised change in volume of the left ventricle (Figure 1.3) over the heart cycle — the *ejection fraction* — is measured from an echocardiography sequence and is used as a coarse indicator of global heart function. In *stress* echocardiography, qualitative local assessment of the myocardium is performed while the patient is at rest and under stress [7]. Wall thickening and abnormal contraction patterns can be indicative of blockages of the coronary arteries (cardiac ischemia) which is a precursor to a heart attack.

2. **Obstetrics** Measurements of the fetal head, abdomen, femur, and heart are made from 2D ultrasound images (Figure 1.2) and compared against population

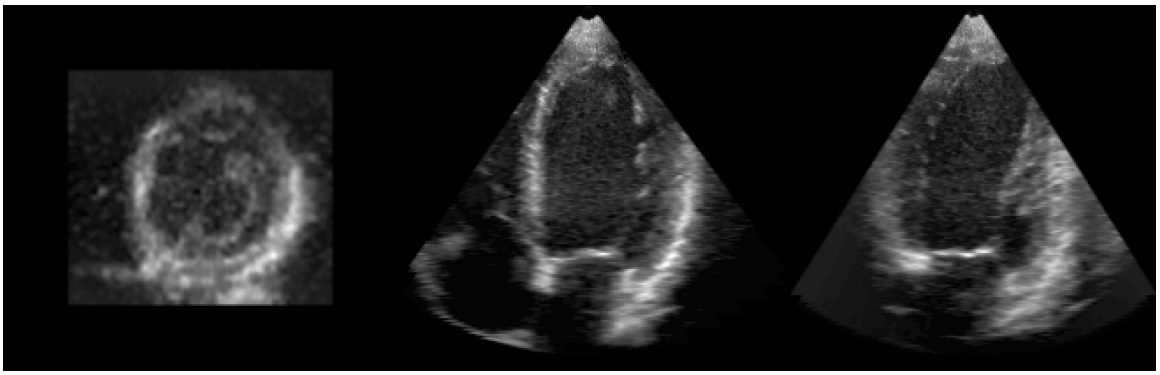


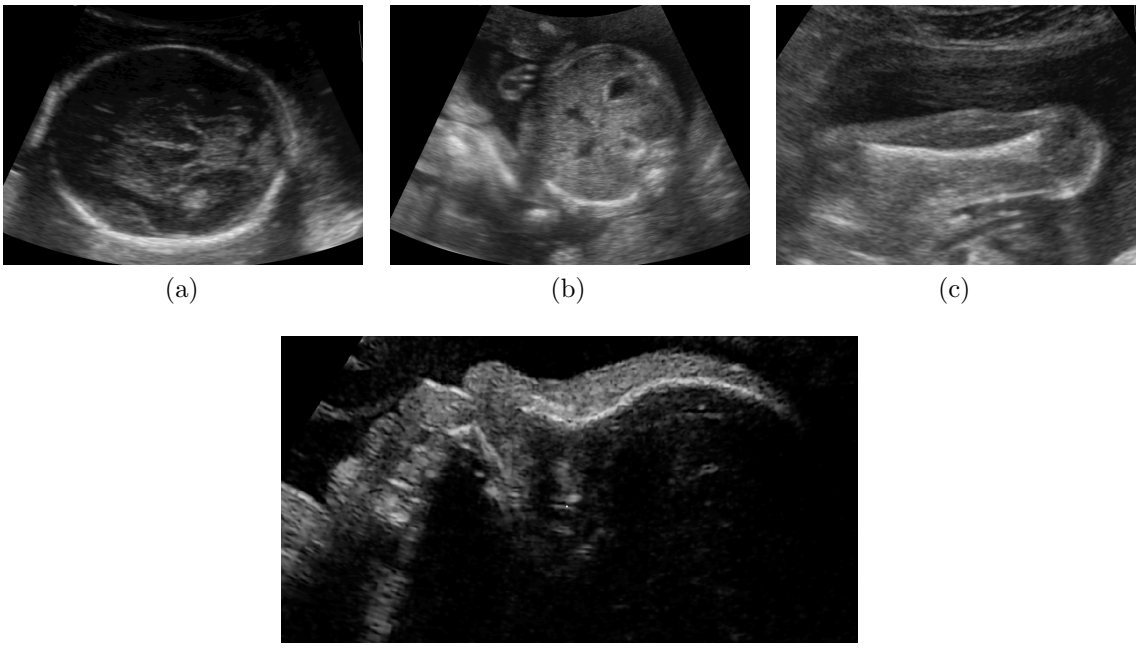
Figure 1.1: Short-axis, four-chamber, and two-chamber views of the left ventricle from a 3D echocardiogram.

growth charts to assess fetal size and detect growth abnormalities [65]. The fetal face is also imaged for visual assessment of congenital defects (Figure 1.2d).

A complete review of diagnostic ultrasound applications is given in [112].

Measuring surface and regional properties of anatomical objects, such as the left ventricle and fetal head, requires accurate segmentation of 2D/3D ultrasound images. This is challenging because of: (a) the lack of visual cues available, (b) missing anatomical boundaries, (c) the inhomogeneity of tissue appearance, and (d) the variation in anatomical shape between subjects. Measuring functional properties (e.g. boundary displacements) is even more complex since it requires consistent segmentation of each frame in the 2D+t/3D+t ultrasound sequence.

An *ideal* segmentation algorithm would, in *real-time*, recover *perfect* geometric representations of an anatomical structure over an image sequence *without* any user interaction (i.e. be fully automatic). For the aforementioned applications, present clinical practice (typically) requires only rough estimates of regional measurements because the derived clinical indices are only interpreted coarsely. Therefore, fully automatic algorithms which perform fast segmentation over short image sequences have been sought after. However, for research applications, more accurate and robust measurements of regional *and* functional properties over *long* image sequences is desirable, and the requirement for real-time processing is secondary since images and sequences can be processed offline. In this thesis, *model-based* segmentation algorithms have been developed which fall into the later category: accurate, robust, and time-agnostic algorithms, which facilitate segmentation of arbitrarily long sequences, have been pursued in favour of immediately simple and fast algorithms. Importantly, the segmentation tasks considered in this thesis are also limited to single anatomical objects of fixed topology.



(a) (b) (c)

(d)

Figure 1.2: Example fetal ultrasound images of the (a) head, (b) abdomen, (c) femur [134], and (d) face.

In the remainder of this chapter, a brief overview of cardiac function is provided, the mathematical notation is defined, and a list of publications is given. The structure of the thesis, from the next chapter onwards, is as follows.

In Chapter 3, a review of 2D+t/3D+t model-based segmentation algorithms is presented. Model representations and definitions for "model fit" are scrutinised, and the distinction between *image-based* and *image-derived* model fit penalties is made.

Next, extensions to 2D+t model-based segmentation are presented. Specifically, a common 2D+t "track-to-last" algorithm, using an explicit model contour (uniform quadratic B-spline) and image-derived fit, is described and the justification for its components, and the assumptions that give rise to them, are examined and removed. The discrete and continuous optimisation algorithms necessary to minimise the proposed extensions are described. The efficacy of the extensions is qualitatively and quantitatively assessed for 2D/2D+t segmentation of simulated edge maps and echocardiography sequences.

In Chapter 4, 2D model-based segmentation is extended to 3D, with Loop and Doo-Sabin subdivision surfaces used for model surface definitions. Three applications of 3D model-based segmentation for ultrasound image analysis are described and presented: skull segmentation for fetal brain image analysis; fetal face segmentation

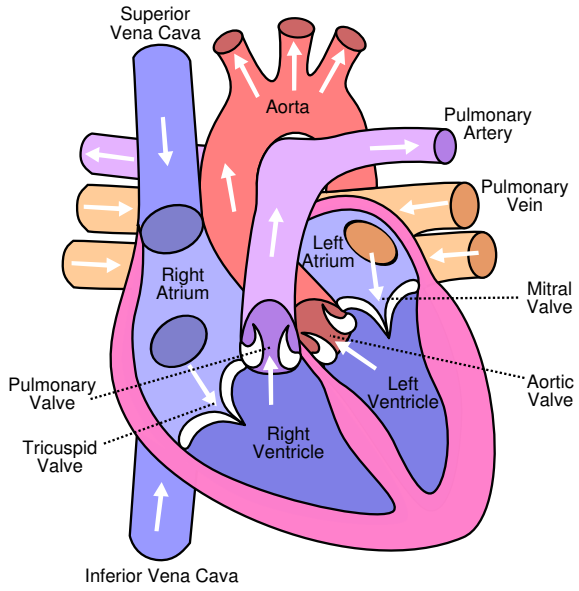


Figure 1.3: Diagram of the human heart [125].

for shape analysis, and single-frame left ventricle segmentation from echocardiograms.

In Chapter 5, a framework to perform model-based segmentation of *multiple* 3D+t sequences while *jointly* optimising an underlying linear basis shape model is presented. The motivating application is right ventricle segmentation from 3D+t echocardiography sequences, where simple surface regularisers are insufficient to accurately interpolate large regions of missing boundary candidates.

In Chapter 6, a method to isolate relevant anatomical boundary positions in an image, using only the structure of edges, is presented. The method is built around a weak parts-based shape model — the Boundary Fragment Model.

Finally, in Chapter 7, the concluding remarks are given and future work is discussed.

1.1 Cardiac Function

The heart is the muscular organ responsible for pumping blood through the body [62]. The heart is divided into four main chambers (Figure 1.3): the left ventricle, left atrium, right ventricle and right atrium. Deoxygenated blood enters the right atrium and is pumped into the right ventricle and then into the lungs through the pulmonary arteries. Oxygenated blood enters through the pulmonary veins into the left atrium and is pumped into the left ventricle and then out through the aortic valve to the aorta.

The wall of the heart is composed of three layers. The *endocardium* is the inner

layer which is in contact with the blood. The *myocardium* is the central layer and responsible for the contraction and relaxation of the ventricles and atria. Finally, the *epicardium* is the outer layer. The *coronary arteries* run along the epicardium and are the only source of blood supply to the myocardium.

Blood is pumped through the heart in a cyclical process consisting of two phases — *systole* and *diastole*. The *systolic* phase is where the ventricles contract to pump blood out of the heart. The *diastolic* phase is where the ventricles relax and the atria contract to pump blood into the heart.

1.2 Notation

In this thesis, matrices are denoted by uppercase letters (X) and vectors with boldface lowercase letters ($\boldsymbol{x}$). Column vectors from matrices are denoted by indexed boldface letters. For example, $\boldsymbol{x}_i$ is column i of the matrix X. Similarly, the j^{th} element of a vector $\boldsymbol{x}$ is denoted by x_j. Unless otherwise stated, column vectors and one-based indexing of matrices and vectors are assumed.

1.3 Publications

Published conference and journal articles in reverse chronological order:

1. J. S. Domingos, **R. V. Stebbing**, P. Leeson, and J. A. Noble. Structured random forests for myocardium delineation in 3D echocardiography. In *MICCAI Workshop on Machine Learning in Medical Imaging*, 2014.

2. J. S. Domingos, **R. V. Stebbing**, and J. A. Noble. Endocardial segmentation using structured random forests in 3D echocardiography. In *MICCAI Challenge on Endocardial Three-dimensional Ultrasound Segmentation*, 2014.

3. A. I. L. Namburete, **R. V. Stebbing**, and J. A. Noble. Diagnostic plane extraction from 3D parametric surface of the fetal cranium. In *MIUA*, pages 27–32, 2014.

4. J. Taylor, **R. V. Stebbing**, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. W. Fitzgibbon. User-specific hand modeling from monocular depth sequences. In *IEEE CVPR*, 2014.

5. S. Rueda, S. Fathima, C. L. Knight, M. Yaqub, A. T. Papageorghiou, B. Rahmatullah, A. Foi, M. Maggioni, A. Pepe, J. Tohka, **R. V. Stebbing**, J. E. McManigle, A. Ciurte, X. Bresson, M. B. Cuadra, C. Sun, G. V. Ponomarev, M. S. Gelfand, M. D. Kazanov, C. Wang, H. Chen, C. Peng, C. Hung, and J. A. Noble. Evaluation and comparison of current fetal ultrasound image

segmentation methods for biometric measurements: A grand challenge. *IEEE TMI*, 2014.

6. A. I. L. Namburete, **R. V. Stebbing**, and J. A. Noble. Cranial parametrization of the fetal head for 3D ultrasound image analysis. In *MIUA*, pages 196–201, 2013.

7. **R. V. Stebbing** and J. A. Noble. Delineating anatomical boundaries using the boundary fragment model. *Medical Image Analysis*, 17(8):1123–1136, 2013.

8. **R. V. Stebbing** and J. E. McManigle. A boundary fragment model for head segmentation in fetal ultrasound. *IEEE ISBI Proceedings of Challenge US*, pages 9–11, 2012.

9. **R. V. Stebbing**, J. E. McManigle, and J. A. Noble. Interpreting edge information for improved endocardium delineation in echocardiograms. In *IEEE ISBI*, pages 238–241, 2012.

10. J. E. McManigle, **R. V. Stebbing**, and J. A. Noble. Modified Hough transform for left ventricle myocardium segmentation in 3-D echocardiogram images. In *IEEE ISBI*, pages 290–293, 2012.

Journal articles under review:

1. **R. V. Stebbing**, A. I. L Namburete, and J. A. Noble. Data-driven shape parameterization for segmentation of the right ventricle from 3D+t echocardiography. *Medical Image Analysis*.

2. A. I. L Namburete, **R. V. Stebbing**, B. Kemp, M. Yaqub, A. T. Papageorghiou, and J. A. Noble. Learning-based prediction of gestational age from ultrasound images of the fetal brain. *Medical Image Analysis*.

Chapter 2

Model-Based Segmentation

2D/3D model-based segmentation enables measurement of global properties (e.g. area, volume) and boundary properties (e.g. perimeter, surface area) of an anatomical object of interest in an image. This is achieved by fitting a geometric description, or *model*, of the anatomical structure — for which regional and boundary properties are straightforward to evaluate — to the image, subject to priors over the model geometry. For 2D+t/3D+t model-based segmentation of an image *sequence*, the purpose is twofold: (a) to recover regional and boundary properties of the anatomical object of interest in all frames, and (b) to facilitate the measurement of functional properties (e.g. boundary displacements, strain) over time. Achieving (a) is straightforward: independent single-frame model-based segmentation of each frame is sufficient. Fulfilling (b), however, requires *anatomically consistent* geometric descriptions so that the same (parametric) position on the model corresponds to the same anatomical position in *all* frames. This is necessary because functional properties depend on the deformation of the geometric description between frames — anatomical consistency ensures that the measurements are faithful to the anatomy being modelled.

Because of its utility, flexibility, and performance, model-based segmentation has had extensive attention in the medical image analysis literature [100, 102]. The number of application-specific frameworks is immense, but many commonalities between segmentation algorithms exist. The primary aim of this chapter is to review 2D/3D model-based segmentation with an emphasis towards ultrasound image analysis applications, with the purpose of consolidating the underlying concepts. First, a review of 2D and 3D model representations is presented, and the differences between *explicit* and *implicit* boundary representations are explained. Second, definitions for "model fit" are scrutinised, and the distinction between *image-based* and *image-derived* model fitting penalties is made. Finally, common sequential 2D+t/3D+t algorithms are briefly revisited as preparation for Chapter 3.

2.1 Explicit and Implicit Model Representations

A model contour (2D) or surface (3D) can describe the anatomical boundary of interest in one of two ways: *explicitly* or *implicitly*.

Explicit Representations An explicit model *directly* describes the geometry of the segmentation boundary by a parametric curve or surface. The simplest explicit model contour or surface definition is a fixed number of discrete points [36, 37, 38, 88]. Importantly, this model is *not* continuous; it is defined at each control point but *not* between. In 2D, a polygonal path — which models straight lines between adjacent control points — is the simplest continuous extension, but B-splines and Bezier curves (as some examples) can also be used when it is necessary to model curved boundaries. Depending on the shape flexibility required by the segmentation task, circles, ellipses and other conic sections may also be appropriate. In 3D, example continuous explicit representations include: triangle or polygonal meshes, (truncated) ellipsoids [116], B-spline and subdivision surfaces [118], Active Geometric Functions (`AGF`) [14, 49, 50], and B-spline Explicit Active Surfaces (`BEAS`) [10].

Implicit Representations Modelling a closed segmentation boundary as the *level-set* of a higher-dimensional function over the image domain is the standard implicit representation used in medical image analysis [27, 119, 136]. Let $\varphi\colon \mathcal{D} \to \mathbb{R}$ denote the *level-set function*, where $\mathcal{D}$ is the image domain and the segmentation boundary is given by the zero level-set $\{\boldsymbol{x}\colon \varphi(\boldsymbol{x}) = 0, \boldsymbol{x} \in \mathcal{D}\}$, and φ is positive (negative) at positions interior (exterior) to the boundary. Typically, φ is initialised as the signed distance transform of the initial segmentation boundary. Importantly, this implicit representation is parameter free and topology free since different topologies of the zero level-set do not imply different topologies of φ [27].

For segmentation of a single object with fixed topology, the primary advantage of a level-set representation over an explicit representation is that defining model fit based on expected properties of the interior and exterior regions — and not *just* the boundary — is *typically* simpler. Region-based model fit definitions have been proposed for explicit model contours (e.g. [82]) but the formulations are awkward. Notable exceptions are `AGF` and `BEAS`, where region-based terms are easily formulated because the model surface is parameterised over spherical or prolate spheroidal coordinates [10, 50].

The level-set representation also has the property that it is *not* intrinsically dependent on the parameterisation of the segmentation boundary [27]. Although this

elegance is intuitively appealing, the absence of any parameterisation makes formulating anatomically consistent 2D+t/3D+t level-set frameworks difficult. Further advantages and disadvantages of each representation are discussed, as appropriate, in the following sections.

2.2 Model Fit Definitions

Definitions for model fit can be broadly separated into two classes: *image-based* and *image-derived* (geometric).

2.2.1 Image-Based Formulations

Image-based penalties define the fit of the model contour or surface *directly* against the image. Description of common frameworks for both explicit and implicit model representations follow.

"Snakes" and Active Appearance Models Assuming an explicit model contour representation, `Snakes` by Kass et al. [78] is probably the most well-known image-based segmentation framework. A spline, parameterised by a finite number of control points, is used for the model contour, and three different image-based penalties are introduced to attract the model contour to lines, edges, and terminations.

Let $\mathcal{I}(\boldsymbol{x})$ denote the image and $\chi(t, X)$ denote the spline contour function parameterised by $t \in [0, 1)$ and the matrix of control points X. The fit of the contour is then given by:

$$E_{\text{FIT}}(X) = \int_{t=0}^{1} \Big\{ \lambda_{\text{LINE}} E_{\text{LINE}}(\chi(t, X)) + \lambda_{\text{EDGE}} E_{\text{EDGE}}(\chi(t, X)) + \lambda_{\text{TERM}} E_{\text{TERM}}(\chi(t, X)) \Big\} \, \mathrm{d}t \tag{2.1}$$

where λ_{LINE}, λ_{EDGE}, and λ_{TERM} are weights. E_{LINE} is given by:

$$E_{\text{LINE}}(\boldsymbol{x}) = \mathcal{I}(\boldsymbol{x}) \tag{2.2}$$

so that depending on the sign of λ_{LINE}, the contour is attracted to either light or dark lines. Similarly, E_{EDGE} is given by:

$$E_{\text{EDGE}}(\boldsymbol{x}) = -|\nabla \mathcal{I}(\boldsymbol{x})| \tag{2.3}$$

so that the model contour is attracted to positions with large image gradients. (Refer to [78] for the definition of E_{TERM}.) In [78], (2.1) is minimised with respect to X using

a gradient descent style optimisation, using a discrete approximation for the integral and finite differences to calculate the first derivatives of E_{LINE}, E_{EDGE}, and E_{TERM}.

For applications where (2.2) and (2.3) are insufficient definitions for model fit, statistical appearance models may be used instead. The simplest is the Active Appearance Model by Cootes et al. [37, 38]. A set of points is used for the model contour, and, given training images with ground-truth model contours, a low-dimensional linear appearance model for the object of interest is learned. The fit of the model contour is measured directly against the image, and optimisation of the model contour *and* the appearance model parameters is performed jointly using gradient descent with a scale-space hierarchy [38], or by learning updates [37].

Level-Set Image-Based Formulations Probably the most well-known level-set image-based segmentation framework is by Chan and Vese [31]. Assume that the segmentation boundary separates foreground and background regions with (unknown) constant intensities $\mathcal{I}_{\text{F}}$ and $\mathcal{I}_{\text{B}}$ respectively. The fit of the model contour is given by the functional:

$$E_{\text{FIT}}\left(\varphi, \mathcal{I}_{\text{F}}, \mathcal{I}_{\text{B}}\right) = \int_{\boldsymbol{x}\in\mathcal{D}} \Big\{ \lambda_{\text{F}} \left(\mathcal{I}\left(\boldsymbol{x}\right) - \mathcal{I}_{\text{F}}\right)^2 \mathcal{H}\left(\varphi\left(\boldsymbol{x}\right)\right) + \lambda_{\text{B}} \left(\mathcal{I}\left(\boldsymbol{x}\right) - \mathcal{I}_{\text{B}}\right)^2 \left(1 - \mathcal{H}\left(\varphi\left(\boldsymbol{x}\right)\right)\right) \Big\} \, \mathrm{d}\boldsymbol{x} \tag{2.4}$$

where λ_{F} and λ_{B} are weights and $\mathcal{H}$ is the Heaviside step function:

$$\mathcal{H}\left(x\right) = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

The Chan and Vese functional is actually a first-order approximation of a Bayesian segmentation formulation of the Mumford and Shah functional [106], assuming Gaussian distributions for the foreground and background regions with unknown means and fixed standard deviations [25]. For ultrasound segmentation specifically, a Rayleigh distribution has been proposed instead by Sarti et al. [135].

A deficiency of the Chan and Vese functional is that it does *not* model non-homogeneous foreground and background regions. In [84], Lankton and Tannenbaum propose Localising Region-Based Active Contours (`LRBAC`) which extend the Chan and Vese functional by formulating model fit over a local neighbourhood. In the simplest

case, (2.4) is modified to:

$$E_{\text{FIT}}(\varphi, \mathcal{I}_{\text{F}}, \mathcal{I}_{\text{B}}) = \int_{\boldsymbol{x}\in\mathcal{D}} \delta(\varphi(\boldsymbol{x})) \int_{\boldsymbol{y}\in\mathcal{D}} \mathcal{B}(\boldsymbol{y}, \boldsymbol{x}) \Big\{ \lambda_{\text{F}} (\mathcal{I}(\boldsymbol{y}) - \mathcal{I}_{\text{F}}(\boldsymbol{x}))^2 \mathcal{H}(\varphi(\boldsymbol{y})) + \lambda_{\text{B}} (\mathcal{I}(\boldsymbol{y}) - \mathcal{I}_{\text{B}}(\boldsymbol{x}))^2 (1 - \mathcal{H}(\varphi(\boldsymbol{y}))) \Big\} \, \mathrm{d}\boldsymbol{y} \, \mathrm{d}\boldsymbol{x} \tag{2.5}$$

where $\delta(t)$ is the Dirac delta function, $\mathcal{I}_{\text{F}}$ and $\mathcal{I}_{\text{B}}$ are spatially-varying, and $\mathcal{B}$ defines the local neighbourhood in a given radius r and is given by:

$$\mathcal{B}(\boldsymbol{y}, \boldsymbol{x}) = \begin{cases} 1, & \|\boldsymbol{x} - \boldsymbol{y}\| < r \\ 0, & \text{otherwise} \end{cases}$$

That is, for each position $\boldsymbol{x}$ on the segmentation boundary (zero level-set), (2.5) measures how the intensity at each position $\boldsymbol{y}$ in the *local* neighbourhood fits with the *local* foreground and background intensities at $\boldsymbol{x}$.

Importantly, although this extension facilitates segmentation of non-homogeneous objects, φ must be initialised accurately to ensure correct segmentation [84]. Extensions to (2.5) using more complex appearance models are also described in [84]. An ultrasound-specific formulation, which uses a Rayleigh distribution to model local foreground and background intensities, has also been proposed by Belaid et al. [13].

Edges may also be used to define model fit for a level-set model contour representation. The standard formulation is the Geodesic Active Contours (`GAC`) framework by Caselles et al. [27]. Let $\chi(s)$ denote the contour corresponding to the zero level-set of φ which is parameterised by arc-length s and has length $L(\chi)$. Also, let $\mathcal{J}$ denote the *inverse edge image* (derived from $\mathcal{I}$) which has low intensities at positions corresponding to edges and high intensities elsewhere. The `GAC` energy is then defined straightforwardly as:

$$E_{\text{FIT}}(\varphi) = \int_{s=0}^{L(\chi)} \mathcal{J}(\chi(s)) \, \mathrm{d}s \tag{2.6}$$

Intuitively, minimising (2.6) with respect to φ is equivalent to finding the contour of minimum *weighted* length, where the weights are given by the inverse edge image.

Learning-based appearance models have also been developed which improve segmentation accuracy for some applications. For ultrasound segmentation specifically, Huang et al. [71] propose the use of *sparse* linear appearance models in conjunction with `AdaBoost` [55] to learn a function to discriminate between interior and exterior regions. Assuming that training images with fitted model contours are available (acquisition of the training data is explained in the next paragraph), *overcomplete* dictionaries *and* sparse representations for image patches for the *local* interior and

exterior regions are first learned using `K-SVD` [3]. A *weak learner* to discriminate between interior and exterior image patches is then defined by assuming that the class of an image patch corresponds to the class of the dictionary which results in the lowest reconstruction error. Next, the training image patches are classified using the weak learner and subsequently resampled based on the classification errors (`AdaBoost`). The entire procedure is repeated for a fixed number of iterations, and a discriminant function which is positive for interior regions and negative for exterior regions is defined as the weighted sum of the weak learners. Applying the discriminant function over an input frame generates a new image which is approximately homogeneous over interior and exterior regions, enabling the use of a local region-based fit similar (but not identical) to (2.5).

For single-frame segmentation, Huang et al. [71] assume that a database of training images with fitted model contours is available to learn the discriminant function. For segmentation of a sequence, Huang et al. instead require an *exact* model contour initialisation in the first frame which is used to initialise the discriminant function. Frames are then segmented sequentially and the discriminant function is updated after each frame is processed. Applications of the framework in 3D, and a stochastic extension of the sequential optimisation of the discriminant function, are presented in [70, 72].

Level-Set Optimisation and Geometric Flows In practice, block-coordinate gradient descent is used to minimise level-set image-based functionals with respect to φ and any other model parameters (e.g. $\mathcal{I}_{\mathrm{F}}$, $\mathcal{I}_{\mathrm{B}}$). The reasons for this are twofold. First, fixing φ and minimising the energy function with respect to each additional model parameter *independently* often leads to closed-form solutions which are appealing and simplify implementation. Second, gradient descent optimisation of φ enables the energy functional to be reinterpreted as a *geometric flow*. That is, by (artificially) parameterising the gradient descent optimisation over time, minimisation of the energy functional is reformulated as a partial differential equation which defines the evolution of φ [27, 136].

Importantly, for the described region-based image penalties, the evolution equation is only non-zero at, or near, the zero level-set. For example, the evolution equation corresponding to (2.4) is:

$$\frac{\partial \varphi}{\partial t}(\boldsymbol{x}) = \delta(\varphi(\boldsymbol{x}))\left(-\lambda_{\mathrm{F}}(\mathcal{I}(\boldsymbol{x}) - \mathcal{I}_{\mathrm{F}})^2 + \lambda_{\mathrm{B}}(\mathcal{I}(\boldsymbol{x}) - \mathcal{I}_{\mathrm{B}})^2\right)$$

This observation motivated the development of *Narrow Band* and *Fast Marching Methods* [136] which *only* evolve φ at positions near the zero level-set, drastically reducing the time taken for optimisation. Following this, Whitaker developed the Sparse Field Algorithm (`SFA`) [155], which maintains multiple lists of positions in the image domain which correspond to the zero level-set and adjacent level-sets. The advantages of the `SFA` algorithm are that it is faster than the combination of Narrow Band and Fast Marching Methods, and that an explicit representation of the model contour is available while the contour is being evolved [83]. The Fast Two-Cycle (`FTC`) algorithm [137] further improves upon `SFA` by: (a) using a discrete approximation of the narrow band that facilitates evolution without solving a partial differential equation, and (b) approximating curvature regularisation using Gaussian filtering.

For the aforementioned algorithms, the evolution *speed* is normalised to less than 0.5 [83, 136] so that the zero level-set traverses all image positions between its initialisation and the local minimum solution. Therefore, an accurate model contour initialisation is necessary because the evolution equation, which is evaluated at all positions, must support evolution in the direction of the desired local minimum.

Summary Examples of image-based formulations for model contour fit have been reviewed for both explicit and implicit model contour representations. For an explicit representation, gradient descent optimisation of the model contour and appearance model parameters is performed to minimise the fitting energy. An accurate model contour initialisation is necessary because: (a) gradient descent converges only to a local minimum, and more subtly, (b) the gradient of the image-based penalty, with respect to the model contour and appearance model parameters, depends *only* on image information local to the current model contour estimate. For an implicit representation, deformation of the level-set function based on geometric flows is necessary for their practical application. However, similar to explicit representations, the use of gradient descent requires an accurate model contour initialisation to ensure correct convergence. This is even more crucial for image-based frameworks which use *only* local image information (e.g. `LRBAC` [84]) or update the appearance model using fitted models from previous segmentations (e.g. [70, 71, 72]).

2.2.2 Image-Derived (Geometric) Formulations

Image-derived penalties define the fit of an explicit model contour or surface *geometrically* to detected boundary positions *instead* of directly against the image.

Total Displacement and Point Distance Minimisation In 2D, let $\chi(t, X)$ denote an explicit contour function parameterised by $t \in [0, 1)$ and matrix of control points X. Also, let $\phi \in \mathbb{R}^{2 \times N_\phi}$ denote a matrix of positions of N_ϕ points describing a boundary of interest. An intuitive definition for the fit of the model contour is the *total displacement*:

$$E_{\text{FIT}}(X) = \sum_{i=1}^{N_\phi} \min_{t_i} \|\boldsymbol{\phi}_i - \chi(t_i, X)\|^2 \tag{2.7}$$

which measures the squared distance of each $\boldsymbol{\phi}_i$ to its closest position on the model contour.

Given an initialisation for X, an obvious (and typical) algorithm for minimising (2.7) is *alternation* [92]. First, for each $\boldsymbol{\phi}_i$, the minimum t_i is solved exactly or approximately, using non-linear optimisation or a closed-form algorithm [68]. Each t_i is known as the *footprint*[1] of $\boldsymbol{\phi}_i$ on χ. Next, given $\{t_i\}$, (2.7) is minimised with respect to X. This alternation algorithm is known as Point Distance Minimisation (`PDM`), and while it is intuitively appealing, it is slow and has been shown to have linear convergence [17].

Joint optimisation of X *and* $\{t_i\}$ has been proposed instead by Speer et al. [140], using the Levenberg-Marquardt algorithm [99] with a sparse QR decomposition to perform efficient non-linear optimisation.

For definitions of χ which are linear in X (e.g. polygonal paths, B-splines), the more efficient *Variable Projection* (`VP`) formulation has also been proposed [21, 59]. The `VP` fitting energy $E_{\text{FIT}}^{\text{VP}}$ is derived as follows. Let $\chi(t, X) = X\boldsymbol{b}(t)$, where $\boldsymbol{b}$ is a vector of (potentially non-linear) basis functions in t. Minimising (2.7):

$$\begin{aligned}
&\min_X \sum_{i=1}^{N_\phi} \min_{t_i} \|\boldsymbol{\phi}_i - \chi(t_i, X)\|^2 = \min_{X, \{t_i\}} \sum_{i=1}^{N_\phi} \|\boldsymbol{\phi}_i - X\boldsymbol{b}(t_i)\|^2 \\
&= \min_{\boldsymbol{x}, \boldsymbol{y}, \{t_i\}} \sum_{i=1}^{N_\phi} \left\{ \left(\phi_i^x - \boldsymbol{b}(t_i)^\top \boldsymbol{x}\right)^2 + \left(\phi_i^y - \boldsymbol{b}(t_i)^\top \boldsymbol{y}\right)^2 \right\} \\
&= \min_{\boldsymbol{t}} \min_{\boldsymbol{x}, \boldsymbol{y}} \left\| \begin{bmatrix} \boldsymbol{\phi}^x \\ \boldsymbol{\phi}^y \end{bmatrix} - \begin{bmatrix} B(\boldsymbol{t}) & 0 \\ 0 & B(\boldsymbol{t}) \end{bmatrix} \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{y} \end{bmatrix} \right\|^2 \\
&= \min_{\boldsymbol{t}} \underbrace{\left\| \left(I - \begin{bmatrix} BB^\dagger(\boldsymbol{t}) & 0 \\ 0 & BB^\dagger(\boldsymbol{t}) \end{bmatrix} \right) \begin{bmatrix} \boldsymbol{\phi}^x \\ \boldsymbol{\phi}^y \end{bmatrix} \right\|^2}_{E_{\text{FIT}}^{\text{VP}}(\boldsymbol{t})}
\end{aligned} \tag{2.8}$$

[1] "Footprint" is used in this chapter for consistency with existing literature but is abandoned in favour of *correspondence* or *preimage* in subsequent chapters.

where $\{t_i\}$ has been collapsed into a single vector $\boldsymbol{t}$, $X = [\boldsymbol{x}\ \boldsymbol{y}]$, $[\boldsymbol{\phi}^x\ \boldsymbol{\phi}^y] = \left[\boldsymbol{\phi}_1\ \ldots\ \boldsymbol{\phi}_{N_\phi}\right]^\top$, $B(\boldsymbol{t}) = \left[\boldsymbol{b}(t_1)\ \ldots\ \boldsymbol{b}\left(t_{N_\phi}\right)\right]^\top$, $B^\dagger(\boldsymbol{t})$ is the Moore-Penrose inverse of $B(\boldsymbol{t})$ [131], and (2.8) follows by replacing $\boldsymbol{x}$ and $\boldsymbol{y}$ with the closed-form (linear) minimum solution. The main advantage of minimising $E_{\text{FIT}}^{\text{VP}}$ instead of the original function (2.7) is that it always converges in fewer iterations, including when the same minimisation algorithm for the full function diverges [59]. A complete discussion regarding the implementation and applications of `VP` is given in [59].

For 3D surface recovery, and subdivision surfaces specifically, `PDM` has been applied by Cheng et al. [34] to fit Catmull-Clark [29] and Loop [93] subdivision surfaces to 3D data points. However, instead of determining the footprint of each data point — which does *not* have a closed form for Catmull-Clark or Loop subdivision surfaces — Cheng et al. sample the model surface at *known* footprint positions and associate the closest data point with the sampled model position. That is, only a subset of *all* valid points — the selection of which depends on the current model surface estimate — are used for updating X. Therefore, the algorithm in [34] is *not* minimising the exact 3D extension of (2.7). Examples of similar approaches include [76, 85, 91, 148].

Hoppe et al. [67] use `PDM` for subdivision surface fitting — which is faithful to the 3D extension of (2.7) — *but* a linear approximation to the model surface is used to approximately solve each footprint. Marinov and Kobbelt [97] apply `PDM` for Loop subdivision surface fitting and compute the true footprints using Gauss-Newton optimisation. However, the optimisation of each footprint is slow because each Gauss-Newton update is limited to updating the footprint within a single Loop patch. In [28], Cashman and Fitzgibbon propose an improved algorithm to update footprints in the setting of *joint* optimisation of the Loop subdivision control vertices *and* footprints. To the best of the author's knowledge, the `VP` method has not been extended to subdivision surfaces.

Tangent Distance Minimisation (Normal Displacement) In 2D, the `PDM` algorithm alternates between solving footprints $\{t_i\}$ and minimising:

$$E_{\text{FIT}}(X \mid \{t_i\}) = \sum_{i=1}^{N_\phi} \|\boldsymbol{\phi}_i - \chi(t_i, X)\|^2 \tag{2.9}$$

with respect to X only. That is, footprint variation is *not* modelled during the optimisation of X.

Let $\nu(t, X)$ denote the unit normal vector on the model contour at t. If minor footprint variations are modelled, then the *normal displacement* can be optimised

instead [18, p. 121]:

$$E_{\text{FIT}}\left(X \mid \{t_i\}\right) = \sum_{i=1}^{N_\phi} \left(\left(\boldsymbol{\phi}_i - \chi\left(t_i, X\right)\right) \cdot \nu\left(t_i, X\right)\right)^2 \tag{2.10}$$

which is a first-order approximant of the squared distance of each data point to the model contour [126].

Let X^0 denote the current estimate of the model contour geometry. Assuming that X does not deviate far from X^0 so that $\nu\left(t, X\right) \approx \nu\left(t, X^0\right)$, (2.10) can then be approximated by:

$$E_{\text{FIT}}\left(X \mid \{t_i\}, X^0\right) = \sum_{i=1}^{N_\phi} \left(\left(\boldsymbol{\phi}_i - \chi\left(t_i, X\right)\right) \cdot \nu\left(t_i, X^0\right)\right)^2 \tag{2.11}$$

which is simpler to minimise (especially for χ linear in X). Minimising (2.11) instead of (2.9) is known as Tangent Distance Minimisation (`TDM`) [66]. When the model contour is close to the data points, minimising this local approximation helps move the model contour to a lower fitting energy *without* explicitly handling joint updates of $\{t_i\}$ and X.

In 2D, `TDM` is used to fit B-spline contours to edges in an image [18, p. 127]. Given an initial model contour, it is sampled uniformly, and for each sample $\chi\left(t_i, X^0\right)$, an image-processing filter is applied along the line perpendicular to the model contour to determine the associated edge position $\boldsymbol{\phi}_i$. The geometry is then updated by minimising (2.11) with respect to X.

Example applications of `TDM` in 3D include registration of 3D data points [32] and Loop subdivision surface fitting (in conjunction with `PDM`) [98]. For ultrasound model-based segmentation in particular, `TDM` has been used for 3D echocardiography left ventricle segmentation. For example, in [116], Orderud uses a truncated ellipsoid for the model surface and an Extended Kalman Filter [9] for the non-linear optimisation algorithm. Orderud et al. extend this framework in [117] by replacing the truncated ellipsoid with a more flexible Doo-Sabin subdivision surface [47].

Squared Distance Minimisation In `TDM`, X is set to minimise the distance of each $\boldsymbol{\phi}_i$ to the *line* passing through $\chi\left(t_i, X\right)$ perpendicular to the *fixed* normal $\nu\left(t_i, X^0\right)$. This approximation is unstable in high curvature regions, motivating the use of a second-order approximant to the squared distance instead [153].

In 2D, let $\rho_i > 0$ denote the curvature radius at $\chi\left(t_i, X^0\right)$, and $d_i = \pm \left\|\boldsymbol{\phi}_i - \chi\left(t_i, X^0\right)\right\|$ denote the *signed* distance from the data point to the model contour which is positive if

it is on the same side as the curvature centre and negative otherwise. The second-order approximant to replace (2.10) is given by the *signed distance error* [126]:

$$E_{\text{FIT}}(X \mid \{t_i\}) = \sum_{i=1}^{N_\phi} \left\{ \frac{d_i}{d_i - \rho_i} \left((\boldsymbol{\phi}_i - \chi(t_i, X)) \cdot \hat{\chi}_t(t_i, X) \right)^2 + \left((\boldsymbol{\phi}_i - \chi(t_i, X)) \cdot \nu(t_i, X) \right)^2 \right\} \tag{2.12}$$

where $\hat{\chi}_t(t, X)$ is the unit tangent vector on the model contour at t. For $d_i = 0$ (i.e. the data point is on the model contour), (2.12) reduces to (2.10). For $d_i \to -\infty$, (2.12) reduces to (2.9). Unfortunately, (2.12) is indefinite for $0 < d_i < \rho_i$. In [128], $d_i / (d_i - \rho_i)$ is replaced with $|d_i| / (|d_i| + \rho_i)$; in [153] it is replaced with $\max(0, d_i / (d_i - \rho_i))$. In both cases the *unified* approximant is first-order for $0 < d_i < \rho_i$.

In [128], B-spline contours are fitted to *ordered* data points by minimising the unified second-order approximant with each $\hat{\chi}_t(t_i, X)$ and $\nu(t_i, X)$ derived from the data points. This is the opposite of `TDM` which fixes $\nu(t_i, X)$ based on the current model contour estimate. Extending [128] to unstructured points, Wang et al.[153] define Signed Distance Minimisation (`SDM`) which assumes $\hat{\chi}_t(t_i, X) \approx \hat{\chi}_t(t_i, X^0)$ and $\nu(t_i, X) \approx \nu(t_i, X^0)$.

The signed distance error can also be formulated in 3D [126] and is used in [127, 128] for fitting cubic B-spline patches. Cheng et al. [34] also apply `SDM` for subdivision surface fitting. However, as with `PDM`, Cheng et al. avoid footprint calculation by sampling the model surface and associating the closest data point with each sampled model position.

Summary In this section, algorithms for fitting an explicit model representation to data points in 2D and 3D have been reviewed. Given only data points representing a boundary of interest, the definition for model fit is straightforward; the complexity arises from the necessary optimisation of both data point footprints *and* model parameters.

Alternating footprint calculation and model parameter optimisation is common. `PDM` is the simplest alternation algorithm; `TDM` and `SDM` are extensions with improved convergence properties. In `TDM`, X is updated by minimising a first-order approximant to the squared distance function; in `SDM`, a second-order approximant is used. In 3D, `TDM` has been used for efficient left ventricle segmentation [116, 117], and `SDM` has been used for Catmull-Clark and Loop subdivision surface fitting [34]. In these

example applications, footprint calculation is avoided by sampling the model surfaces and utilising only a subset of all valid boundary points.

Joint optimisation of footprints and model parameters has also been proposed. In 2D, for definitions of χ which are linear in the model parameters, the more efficient `VP` formulation has also been developed which eliminates the model parameters from the optimisation altogether. In 3D, joint optimisation for Loop subdivision surface fitting has been developed by Cashman and Fitzgibbon [28] by extending the footprint optimisation algorithm of Marinov and Kobbelt [97].

In comparison to image-based frameworks, the advantage of image-derived (geometric) model fit penalties with explicit model representations is that optimisation methods more powerful than gradient descent can be used. Additionally, although optimisation of model parameters and data point footprints is difficult and not as elegant as level-set geometric flows, it will be shown that retaining the explicit parameterisation for the segmentation boundary enables straightforward formulation of anatomically consistent 2D+t/3D+t segmentation algorithms.

2.3 Sequential Segmentation

For 2D+t/3D+t model-based segmentation, it is desirable to recover a geometric description of the segmentation boundary for all images (frames) in a sequence. For an anatomically consistent segmentation, positions on the *explicit* model contour or surface must correspond to the same semantic or anatomical boundary positions throughout the sequence. The purpose of this section is to briefly review the most common approach employed for ultrasound sequence segmentation: *sequential* 2D+t/3D+t segmentation.

The first-order "track-to-last" algorithm is the simplest method for applying a 2D/3D segmentation framework to an image sequence. First, the model is fitted to the first frame of the sequence. Next, the fitted model is used as a *prior* [18, p. 188] and/or initialisation (e.g. [70, 116]) for the model in the next frame. This frame is subsequently segmented, and the entire process is repeated for the entirety of the image sequence. To model *known* frame-to-frame motion or drift, deterministic transformations of the fitted model can be used [18, p. 193], and second-order autoregressive models — which use the fitted models from two previous frames — are also suitable when constant acceleration is assumed [18, p. 204]. Importantly, for implicit model representations, the segmentations are *not* anatomically consistent because the segmentation boundary is *not* explicitly parameterised (by definition).

For a *linear* dynamical system where the *state* and *output* vectors are corrupted by Gaussian noise, the Kalman filter is a minimum squared error state estimator [113] and is commonly used for sequential model-based segmentation. In particular, for explicit model representations with χ linear in X, the Kalman filter has been used for 2D+t [18, p. 218] and 3D+t [116, 117] regularised sequential `TDM`:

- In [18, p. 218], with X^0 denoting the *predicted* control point positions for the current frame and $\chi(t, X) = X\boldsymbol{b}(t)$, the Kalman filter state is the vector of control point position differences: $X - X^0$. For each data point, the Kalman filter output is the projection of the displacement of the corresponding model contour position along the contour normal: $((X - X^0)\,\boldsymbol{b}(t_i)) \cdot \nu(t_i, X^0)$.

- In [116], the state is the vector of 10 deformation parameters specifying the translation, scaling, rotation and bending of a truncated ellipsoid model. The Kalman filter output vector is the projection of the deformation Jacobian along the surface normal.

- In [117], the state comprises of rigid transformation parameters and constrained control vertex displacements which specify the geometry of a Doo-Sabin subdivision surface. Similar to [116], the Kalman filter output vector is the projection of the deformation Jacobian along the surface normal.

Importantly, the Kalman filter has also been used for 3D+t regularised sequential `PDM` for anatomically consistent left ventricle segmentation from echocardiography sequences [118].

For dynamical systems where the state and output vectors are corrupted by non-Gaussian noise, the stochastic `CONDENSATION` algorithm has been developed [74, 75]. The `CONDENSATION` algorithm is an example of a *particle filter*, which is itself a *Sequential Monte Carlo* method [48] applied to a Hidden Markov Model [16, p. 645]. Identical to the Kalman filter, the `CONDENSATION` algorithm performs sequential segmentation of an image sequence. However, instead of propagating a mean model contour and state covariance matrix, the `CONDENSATION` algorithm maintains a set of sample model contours which approximate the posterior distribution of the model contour. Further details, related algorithms, and example applications are given in [120].

To conclude, the advantages of sequential segmentation algorithms are that: (a) they are straightforward to implement, and (b) facilitate online segmentation. The principal disadvantage is that segmentation errors and failures accumulate over time, making successful segmentation of long sequences difficult.

2.4 Conclusions

In this chapter, 2D/3D model-based segmentation has been reviewed. Definitions for implicit and explicit model representations have been given, and the distinction between image-based and image-derived (geometric) model fitting penalties has been made. Common sequential 2D+t/3D+t model-based segmentation formulations and algorithms have also been briefly reviewed.

The optimisation algorithms for each representation and model fitting penalty have been precisely described. For image-based penalties, independent of model representation, gradient descent optimisation is typical and an accurate model initialisation is necessary. For image-derived penalties with explicit model representations, optimisation of both data point footprints and model parameters is formally necessary, but optimisation methods more powerful than gradient descent are also suitable. Historically, alternating footprint and model parameter optimisation has been common, with improvements to the latter achieved using first- and second-order approximants to the squared distance function. Crucially, footprint optimisation has also been avoided altogether in some applications by sampling fixed positions on the model contour or surface and selecting only a subset of valid data points perpendicular to the model.

In the next chapter it will be shown, in 2D, that alternation can be abandoned in favour of joint optimisation, and restricting boundary candidates to be perpendicular to the model contour is unnecessary. Furthermore, improved 2D+t formulations superior to sequential optimisation will be presented. In latter chapters, joint optimisation for subdivision surface fitting in 3D will then be described in detail, enabling straightforward specification of frameworks for various ultrasound segmentation tasks.

Chapter 3

2D+t Model-Based Segmentation

In this chapter, extensions to 2D/2D+t model-based segmentation are presented. To start, a common 2D+t "track-to-last" algorithm is described and the justifications for its components, and the assumptions that give rise to them, are examined and removed.

First, restricting boundary candidates to be perpendicular to the model contour is shown to be necessary only if, for each boundary candidate, its corresponding position on the model contour is fixed. If model correspondences are optimised jointly with the model contour, this restriction is redundant. Additionally, coherent boundary candidate selection can be further encouraged by introducing pairwise constraints and abandoning independent boundary candidate selection altogether.

Second, sequential processing of frames is shown to be necessary because anatomical consistency is *not* explicitly modelled by the energy which the "track-to-last" algorithm minimises. That is, anatomical consistency arises only as a result of the constrained, precise, sequential algorithm. As replacements, two formulations for 2D+t segmentation are described which model anatomical consistency explicitly. Both formulations are amenable to joint optimisation, facilitating segmentation of arbitrarily long sequences and enabling user-interaction of the model contour at any point in the sequence.

Following this, the discrete and continuous optimisation algorithms necessary to minimise the proposed extensions are described. Finally, the efficacy of the proposed extensions are qualitatively and quantitatively assessed for 2D/2D+t segmentation of simulated edge maps and echocardiography sequences.

3.1 "Track-to-last" 2D+t Segmentation

In this section, a representative and typical "track-to-last" framework for 2D+t model-based segmentation is presented and described. The purpose of this section is to demonstrate that the "track-to-last" algorithm is in fact constrained sequential minimisation of a general energy function. First, single-frame boundary candidate selection and model contour fitting are described. Frame-to-frame fitting is detailed next, followed by a description of the complete algorithm and its global formulation.

Importantly, the "track-to-last" formulation presented in this section uses an explicit model contour (§2.1) and an image-derived model fit penalty (§2.2.2). An explicit model contour is used instead of an implicit representation because the direct parameterisation of the segmentation boundary makes anatomical consistency straightforward to formulate. An image-derived (geometric) model fit is used because it avoids direct continuous optimisation over the image data, resulting in fewer local minima and enabling the use of non-linear continuous optimisation algorithms more powerful than gradient descent. Instead, anatomical consistency is enforced during the discrete optimisation step of boundary candidate selection.

3.1.1 Input Data

Let F denote the number of frames to segment. The input data to the 2D+t segmentation algorithm is a set of images $\left\{\mathcal{I}^f\right\}_{f=1}^{F}$ and the corresponding set of boundary candidates $\left\{\phi^f\right\}_{f=1}^{F}$. Each $\phi^f \in \mathbb{R}^{2\times N_\phi^f}$ is the matrix of N_ϕ^f boundary candidate positions in frame f. It is assumed that the boundary candidates have been generated by applying an off-the-shelf edge detector or similarly purposed algorithm.

3.1.2 Model Contour

Let $X \in \mathbb{R}^{2\times N_X}$ denote the matrix of N_X control points which define the geometry of the open or closed parametric continuous model contour. Positions on the model contour are defined by the function $\chi(u, X)$, where $u \in \Omega$ is a contour coordinate and $\Omega \subset \mathbb{R}$ is the contour domain. Normal vectors — or approximate normal vectors — on the model contour are defined by the function $\nu(u, X)$. The exact definitions of χ and ν are unimportant and can be given by a polygonal path, B-spline, Bezier curve, or any other continuous geometric curve of interest.

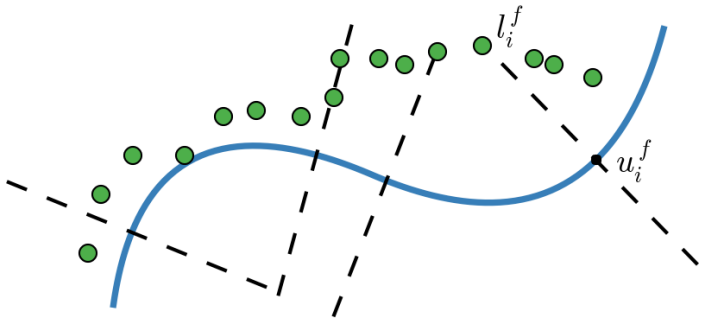


Figure 3.1: Each l_i^f is set to the index of the closest boundary candidate (green) that is perpendicular to the model contour (blue) at coordinate u_i^f.

3.1.3 Single-Frame Fit

Let $X^f \in \mathbb{R}^{2\times N_X}$ denote the matrix of control points which define the model contour in frame f and assume an initialisation is available. To define the fit of the model contour, $N_{\boldsymbol{u}}^f$ points along the contour are first sampled at coordinates specified by the vector $\boldsymbol{u}^f \in \Omega^{N_{\boldsymbol{u}}^f}$. Each element of $\boldsymbol{u}^f$ is referred to as a model coordinate or *correspondence*. Typically, $\boldsymbol{u}^f$ is set so that samples are uniformly distributed across the length of the contour [18].

For each sampled coordinate on the model contour, indexed by i, its associated boundary candidate index l_i^f — or *label* — is given by:

$$l_i^f = \underset{l\in\Upsilon^f\left(u_i^f, X^f\right)}{\arg\min} \left\| \boldsymbol{\phi}_l^f - \chi\left(u_i^f, X^f\right)\right\|^2 \tag{3.1}$$

where Υ^f is a function which returns the set of indices of boundary candidates in frame f which are perpendicular to the model point at coordinate u_i^f. In other words, each l_i^f is set to the index of the closest boundary candidate perpendicular to the model point at u_i^f (Figure 3.1). To simplify further discussion, assume that all correspondences with no boundary candidate sufficiently close to the model contour are removed from $\boldsymbol{u}^f$. In practice this is achieved by removing the entries in $\boldsymbol{u}^f$ for which the minimum squared distance in (3.1) exceeds a specified threshold.

With $\boldsymbol{l}^f$ denoting the vector of boundary candidate indices, the fit of the model contour is given by:

$$E^f\left(X^f \mid \boldsymbol{u}^f, \boldsymbol{l}^f\right) = \underbrace{\sum_{i=1}^{N_{\boldsymbol{u}}^f} \left\| \boldsymbol{\phi}_{l_i^f}^f - \chi\left(u_i^f, X^f\right)\right\|^2}_{E_{\text{Fit}}^f\left(X^f \mid \boldsymbol{u}^f, \boldsymbol{l}^f\right)} + \lambda_{\mathcal{R}}\mathcal{R}\left(X^f\right) \tag{3.2}$$

where $\mathcal{R}$ is the regulariser over the model contour and $\lambda_{\mathcal{R}}$ is a constant that controls its weight. X^f is found by minimising (3.2) with respect to X^f only ($\boldsymbol{u}^f$ and $\boldsymbol{l}^f$ are fixed). Exact minimisers for X^f can be found in special cases, e.g. when χ is linear in X^f, and $\lambda_{\mathcal{R}} = 0$ or $\mathcal{R}$ is quadratic in X^f [18].

In practice, the entire procedure — setting $\boldsymbol{u}^f$, constructing $\boldsymbol{l}^f$, and refining X^f — is repeated for a fixed number of iterations.

3.1.4 Frame-to-Frame Fit

Assume that X^{f+1} is initialised by a deterministic (typically linear) transformation of X^f, with $\boldsymbol{u}^{f+1}$ and $\boldsymbol{l}^{f+1}$ subsequently set following §3.1.3. Now, let $\tilde{l}_i^f$ denote the index of the boundary candidate in frame f closest to the model point at coordinate u_i^{f+1}:

$$\tilde{l}_i^f = \arg\min_{\tilde{l}} \left\| \boldsymbol{\phi}_{\tilde{l}}^f - \chi\left(u_i^{f+1}, X^f\right) \right\|^2 \tag{3.3}$$

To enforce anatomical consistency, each l_i^{f+1} is adjusted so that the local appearance of its selected boundary candidate in frame $f+1$ is similar to that of the boundary candidate indexed by $\tilde{l}_i^f$ in frame f. This is standard block matching [118].

Let $\Gamma^f(l)$ denote the function which returns the set of indices of boundary candidates within a fixed radius or window of boundary candidate l in frame f. The i^{th} updated boundary candidate index in frame $f+1$ is given by:

$$l_i^{f+1} \leftarrow \arg\min_{l \in \Gamma^{f+1}\left(l_i^{f+1}\right)} \mathcal{A}\left(\boldsymbol{\vartheta}_l^{f+1}, \boldsymbol{\vartheta}_{\tilde{l}_i^f}^f\right) \tag{3.4}$$

where $\boldsymbol{\vartheta}_l^f$ denotes the appearance vector associated with boundary candidate l in frame f and is derived from $\mathcal{I}^f$, and $\mathcal{A}$ is a function which measures appearance difference. The simplest definition for $\boldsymbol{\vartheta}_l^f$ is the vector of intensities of a square patch of fixed size centred at the boundary candidate l [19]. Common definitions for $\mathcal{A}$ used in ultrasound image analysis, and "speckle tracking" in particular, are Sum of Squared Differences (SSD) [156, 157] and Sum of Absolute Differences (SAD) [57, 61, 118]. In practice, SSD is most appropriate when an assumption of zero-mean Gaussian noise is reasonable, whereas SAD is more appropriate when outliers are expected.

With $\boldsymbol{l}^{f+1}$ updated, X^{f+1} is refined by minimising $E^{f+1}\left(X^{f+1} \mid \boldsymbol{u}^{f+1}, \boldsymbol{l}^{f+1}\right)$. As in §3.1.3, the entire procedure is typically repeated to further refine X^{f+1}.

3.1.5 "Track-to-last" Algorithm

With the algorithms for single-frame and frame-to-frame fitting defined, the "track-to-last" algorithm is defined as follows:

S.1 Assuming X^1 is available either by automatic initialisation or user input, X^1 is refined using the single-frame fitting algorithm described in §3.1.3.

S.2 X^2 (X^{f+1}) is initialised by a deterministic transformation of X^1 (X^f) and then refined using the frame-to-frame fitting algorithm described in §3.1.4. This is repeated until all frames have been segmented.

The output of the "track-to-last" algorithm is the set of matrices $\left\{X^f\right\}_{f=1}^{F}$ which describe the anatomically consistent model contours in each frame ($\left\{\boldsymbol{u}^f\right\}_{f=1}^{F}$ and $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F}$ are discarded).

3.1.6 "Track-to-last" Global Formulation

Consider summing all of the fit, regularisation, and appearance terms included in the "track-to-last" sequential algorithm. A single global energy over all of the unknowns — $\left\{X^f\right\}_{f=1}^{F}$, $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F}$, and $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F}$ — can be written:

$$\begin{aligned} E_{\textsc{TrackToLast}}\left(\left\{X^f\right\}_{f=1}^{F}, \left\{\boldsymbol{u}^f\right\}_{f=1}^{F}, \left\{\boldsymbol{l}^f\right\}_{f=1}^{F}\right) = & \sum_{f=1}^{F} E_{\textsc{Fit}}^{f}\left(X^f \mid \boldsymbol{u}^f, \boldsymbol{l}^f\right) \\ & + \sum_{f=1}^{F-1} \sum_{i=1}^{N_{\boldsymbol{u}}^{f+1}} \lambda_{\mathcal{A}} \mathcal{A}\left(\boldsymbol{\vartheta}_{l_i^{f+1}}^{f+1}, \boldsymbol{\vartheta}_{\tilde{l}_i^f}^{f}\right) \\ & + \sum_{f=1}^{F} \lambda_{\mathcal{R}} \mathcal{R}\left(X^f\right) \end{aligned} \tag{3.5}$$

where $\tilde{l}_i^f$ has its definition from (3.3), and $\lambda_{\mathcal{A}}$ is a constant that weights the $\mathcal{A}$ terms.

With reference to (3.5) and §3.1.5 it can now be shown that the sequential "track-to-last" algorithm is in fact just one particular algorithm for initialising the unknowns and minimising components of $E_{\textsc{TrackToLast}}$.

To start, **S.1** initialises $\boldsymbol{u}^1$ from X^1 and $\boldsymbol{l}^1$ is set by minimising (3.5) subject to the constraint that each selected boundary candidate be perpendicular to the model contour (3.1). X^1 is then refined by minimising (3.5) with respect to X^1 only — $\boldsymbol{u}^1$ is fixed.

Next, **S.2** initialises X^2 (X^{f+1}) from X^1 (X^f) and sets $\boldsymbol{u}^2$ ($\boldsymbol{u}^{f+1}$) accordingly. $\boldsymbol{l}^2$ ($\boldsymbol{l}^{f+1}$) is then set by sequential minimisation of terms in (3.5). First, $\boldsymbol{l}^2$ is set by minimising the E^2_{FIT} term, again constrained so that each selected boundary candidate is perpendicular to the model contour. Then, $\boldsymbol{l}^2$ is refined by minimising each of the $\mathcal{A}\left(\boldsymbol{\vartheta}^2_{l^2_i}, \boldsymbol{\vartheta}^1_{l^1_i}\right)$ terms, subject to the constraint that each l^2_i does not deviate outside of a fixed radius or window.

In summary, the "track-to-last" algorithm of §3.1.5 minimises (3.5) by optimising each X^f sequentially and separately. In addition:

1. Each $\boldsymbol{u}^f$ is initialised after each update of X^f but is *not* a free variable in the optimisation of (3.5) — each $\boldsymbol{u}^f$ is *fixed*.

2. Each $\boldsymbol{l}^f$ is set by sequential *constrained* minimisation of individual terms in (3.5) — it is *not* the unconstrained joint minimum of the fit and appearance terms.

3.2 Generalised 2D+t Segmentation

In §3.1.6 it was shown that the "track-to-last" algorithm minimises (3.5) by sequential optimisation of X^f, while initialising and fixing $\boldsymbol{u}^f$ and constraining the selection of $\boldsymbol{l}^f$ at each step. In this section, the justification and motivation for these constraints is first presented. Next, it is shown that the constraints on $\boldsymbol{l}^f$ can be removed when $\boldsymbol{u}^f$ are treated as *free* parameters and optimised with X^f. Two formulations to enable joint 2D+t segmentation are then presented. Finally, the introduction of pairwise terms over boundary candidate indices is described.

3.2.1 Why is Perpendicular Boundary Candidate Selection Necessary?

Let $\chi(u)$ and $\psi(u)$ denote two parametric contours ($u \in \Omega$) which describe two curves with similar geometry but different parameterisations (Figure 3.2a). The parameters controlling the geometry of both curves are omitted to simplify discussion. Additionally, let $\boldsymbol{u} \in \Omega^{N_{\boldsymbol{u}}}$ denote a vector of $N_{\boldsymbol{u}}$ samples from Ω.

To establish a measure of difference from χ to ψ, a simple approach is to use the average squared distance between both contours evaluated at each coordinate in $\boldsymbol{u}$:

$$d_{\text{NAÏVE}}(\chi, \psi) = \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \|\psi(u_i) - \chi(u_i)\|^2$$

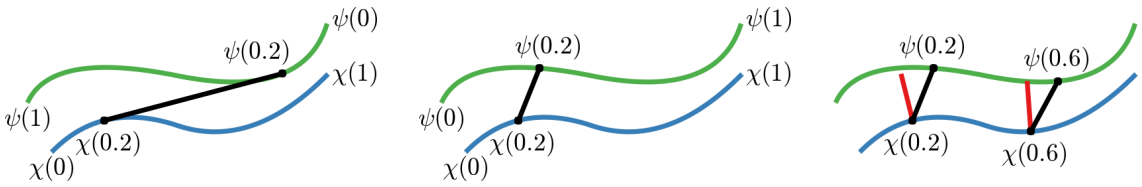


Figure 3.2: (a) χ (blue) and ψ (green) have similar geometry but different parameterisations. Distance between points at the same coordinate (black) overestimate geometric difference ($d_{\text{NAÏVE}}$). (b) χ (blue) and ψ (green) with similar geometry *and* similar parameterisations. (c) Normal displacements (red) discard tangential displacements to estimate geometric distance (d_{NORMAL}).

The shortcoming of this definition is that differences in parameterisation result in overestimation of the geometric distance between the curves (Figure 3.2a) [18, p. 59].

An alternative approach is to calculate the average squared *normal* displacement between the contours (Figures 3.2b, 3.2c):

$$d_{\text{NORMAL}}(\chi, \psi) = \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \left[(\psi(u_i) - \chi(u_i)) \cdot \nu(u_i) \right]^2 \tag{3.6}$$

where $\nu(u)$ gives the unit normal vector of χ at coordinate u. d_{NORMAL} can be shown to be optimal when the curves are close and the difference in parameterisation of the contours is small [18, pp. 120–122]. Intuitively, d_{NORMAL} discards the tangent displacement because it is assumed that it is *only* as a result of the difference in parameterisation between χ and ψ.

Now consider measuring the distance of χ to a set of boundary candidates ϕ (Figure 3.3a). Following [18], a subset of points in ϕ are treated as *images* of an *unknown* parametric contour ψ. Since ψ is unknown, $\psi(u_i)$ is unavailable and (3.6) cannot be evaluated. To overcome this, [18, p. 127] proposes sampling χ and selecting boundary candidates from ϕ which are close to each model point. In doing so it is assumed that each selected boundary candidate corresponds *exactly* to the model point at the *fixed* coordinate u_i. In other words, the *implied* ψ from ϕ is parameterised identically to χ (Figure 3.3b). Therefore, each boundary candidate is restricted to be perpendicular to $\chi(u_i)$ so that its tangential displacement — which is assumed in the formulation of d_{NORMAL} to arise *only* from a difference in parameterisation — is necessarily zero.

In summary, perpendicular boundary candidate selection — as defined in §3.1.3 and (3.1) — is necessary when parameterisation differences between χ and the selected boundary candidates from ϕ, the implied ψ, are *not* modelled.

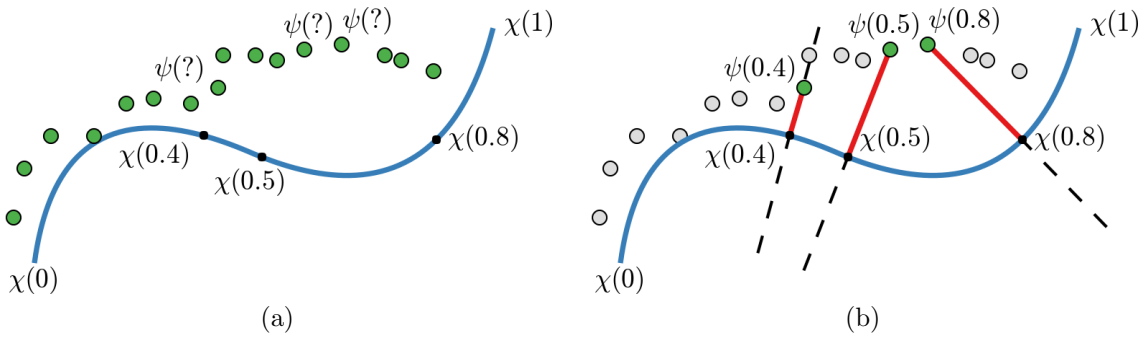


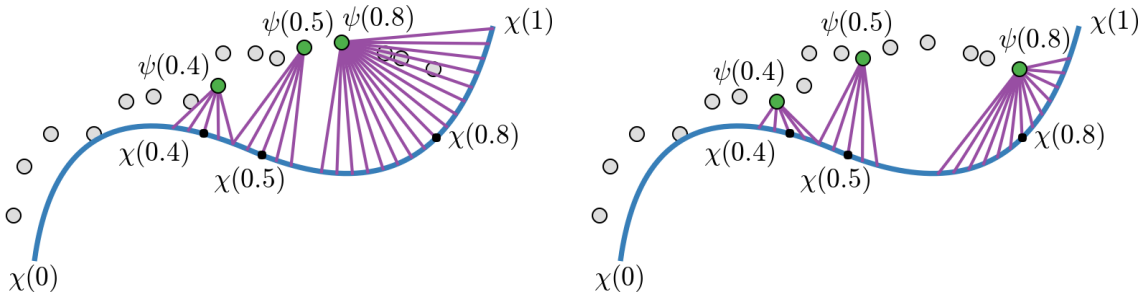
Figure 3.3: (a) χ (blue) and boundary candidates (green). (b) Boundary candidates perpendicular to samples on χ are selected as corresponding points on the *implied* ψ. d_{Normal} is measured by averaging the squared length of the normal displacements (red). Unselected boundary candidates are shown in gray.

3.2.2 Average Minimum Squared Distance

Assume that the selected boundary candidates $\{\psi(u_i)\}_{i=1}^{N_{\boldsymbol{u}}}$ from §3.2.1 are available. The average minimum squared distance of χ to $\{\psi(u_i)\}_{i=1}^{N_{\boldsymbol{u}}}$ is defined as:

$$d_{\text{Min}}\left(\chi, \{\psi(u_i)\}_{i=1}^{N_{\boldsymbol{u}}}\right) = \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \min_{t \in \Omega} \|\psi(u_i) - \chi(t)\|^2 \tag{3.7}$$

For each $\psi(u_i)$, the distance to χ is taken as the minimum over *all* possible values for the model coordinate t (Figure 3.4a). Unlike $d_{\text{Naïve}}$ and d_{Normal}, d_{Min} does *not* assume directly or implicitly that χ is in exact or close parametric correspondence to the samples of ψ. For d_{Min}, it is therefore acceptable to select boundary candidates *without* the restriction that they are perpendicular to χ (Figure 3.4b).



(a) Perpendicular boundary candidates.

(b) Closest boundary candidates.

Figure 3.4: d_{Min} computes the minimum squared distance of each sample to χ. Multiple correspondences (purple) for each boundary candidate represent $\min_{t \in \Omega}$. d_{Min} does *not* require boundary candidates to be perpendicular to χ.

3.2.3 Average Minimum Squared Distance Contour Fit

Let X denote the matrix of parameters which control the geometry of χ. Using (3.7), the average minimum squared distance fit of χ is:

$$E_{\text{MIN}}(X) = \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \min_{t \in \Omega} \|\psi(u_i) - \chi(t, X)\|^2 \tag{3.8}$$

Direct minimisation of (3.8) is difficult because the presence of the min in the summation complicates evaluation of derivatives. Making the variable change $t \to t_i$:

$$E_{\text{MIN}}(X) = \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \min_{t_i \in \Omega} \|\psi(u_i) - \chi(t_i, X)\|^2 \tag{3.9}$$

motivates an Iterated Closest Point (`ICP`) [15, 54] style algorithm which minimises E_{MIN} by performing the following steps in alternation until convergence:

1. Fix X and solve for $\{t_i\}_{i=1}^{N_{\boldsymbol{u}}}$ by independently minimising each summand.
2. Fix $\{t_i\}_{i=1}^{N_{\boldsymbol{u}}}$ and optimise X.

Both steps reduce the energy, and it is bounded below, so convergence to a local minimum is guaranteed.

While `ICP` is sufficient to minimise (3.9), E_{MIN} can be reformulated altogether so that it does not contain the min in the summation. Consider minimising E_{MIN} with respect to X:

$$\begin{aligned}
\min_X E_{\text{MIN}}(X) &= \min_X \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \min_{t_i \in \Omega} \|\psi(u_i) - \chi(t_i, X)\|^2 \\
&= \min_X \min_{t_1 \in \Omega} \dots \min_{t_{N_{\boldsymbol{u}}} \in \Omega} \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \|\psi(u_i) - \chi(t_i, X)\|^2 \\
&= \min_X \min_{\boldsymbol{t}} \underbrace{\frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \|\psi(u_i) - \chi(t_i, X)\|^2}_{\tilde{E}_{\text{MIN}}(X, \boldsymbol{t})}
\end{aligned} \tag{3.10}$$

where $\{t_i\}_{i=1}^{N_{\boldsymbol{u}}}$ has been collapsed into a single vector $\boldsymbol{t}$. As shown, minimising E_{MIN} with respect to X is equivalent to minimising $\tilde{E}_{\text{MIN}}$ with respect to X *and* $\boldsymbol{t}$. Importantly, this reformulation does not change the minimum.

In summary, minimising $\tilde{E}_{\text{MIN}}$ with respect to X and $\boldsymbol{t}$ is equivalent to fitting χ to $\{\psi(u_i)\}_{i=1}^{N_{\boldsymbol{u}}}$ using d_{MIN}. Therefore, $\tilde{E}_{\text{MIN}}$ inherits the properties of modelling parameterisation differences between χ and $\psi(u_i)$: $\psi(u_i)$ is *not* required to be perpendicular to χ.

3.2.4 Free Correspondences and Unconstrained Boundary Candidate Selection

Consider E^f_{FIT} as introduced in (3.2):

$$E^f_{\text{FIT}}\left(X^f \mid \boldsymbol{u}^f, \boldsymbol{l}^f\right) = \sum_{i=1}^{N^f_{\boldsymbol{u}}} \left\| \boldsymbol{\phi}^f_{l^f_i} - \chi\left(u^f_i, X^f\right) \right\|^2$$

and $\tilde{E}_{\text{MIN}}$ as introduced in (3.10):

$$\tilde{E}_{\text{MIN}}\left(X, \boldsymbol{t}\right) = \frac{1}{N_{\boldsymbol{u}}} \sum_{i=1}^{N_{\boldsymbol{u}}} \left\| \psi\left(u_i\right) - \chi\left(t_i, X\right) \right\|^2 \tag{3.11}$$

Ignoring the normalisation by $N_{\boldsymbol{u}}$ in (3.11), E^f_{FIT} and $\tilde{E}_{\text{MIN}}$ are identical in form — each selected boundary candidate $\boldsymbol{\phi}^f_{l^f_i}$ is equivalent to the sample $\psi\left(u_i\right)$, and the vector of correspondences $\boldsymbol{u}^f$ is equivalent to the vector of coordinates $\boldsymbol{t}$.

The only difference between E^f_{FIT} and $\tilde{E}_{\text{MIN}}$ is that the correspondences $\boldsymbol{u}^f$ are *fixed* when optimising E^f_{FIT} in the "track-to-last" algorithm (§3.1.6) whereas $\tilde{E}_{\text{MIN}}$ is minimised with respect to X *and* $\boldsymbol{t}$ (§3.2.3). Therefore, if each $\boldsymbol{u}^f$ is treated as a free parameter when minimising (3.5), then E^f_{FIT} inherits identical properties to $\tilde{E}_{\text{MIN}}$ and the requirement that $\boldsymbol{l}^f$ be restricted to boundary candidates perpendicular to the model contour can be removed.

In §3.1.6 it was shown that the "track-to-last" algorithm performs *sequential* and *constrained* minimisation of individual terms of (3.5) with respect to each $\boldsymbol{l}^f$. As discussed, the first constraint was that each l^f_i be perpendicular to the model contour (3.1) when optimising E^f_{FIT} — this is not required if $\boldsymbol{u}^f$ is optimised with X^f. The second constraint was that each l^f_i be refined in a limited window (3.4) when optimising $\mathcal{A}\left(\boldsymbol{\vartheta}^{f+1}_{l^{f+1}_i}, \boldsymbol{\vartheta}^f_{\tilde{l}^f_i}\right)$. This hard constraint can be removed and a term to penalise spatial distance of boundary candidates in adjacent frames can be used instead:

$$E_{\text{G}}\left(\boldsymbol{\phi}^{f+1}_{l^{f+1}_i}, \boldsymbol{\phi}^f_{\tilde{l}^f_i}\right) = \left\| \boldsymbol{\phi}^{f+1}_{l^{f+1}_i} - \boldsymbol{\phi}^f_{\tilde{l}^f_i} \right\|^2$$

With the constraints on each l^f_i removed, the sequential optimisation of E^f_{FIT} and $\mathcal{A}$ is no longer required.

In summary, if E^f_{FIT} is minimised with respect to $\boldsymbol{u}^f$ and X^f, then $\boldsymbol{l}^f$ can be set to the *unconstrained* minimum of the sum of E^f_{FIT}, $\mathcal{A}$ and E_{G} terms.

Additionally, an important consequence of unconstrained minimisation with respect to l^f_i is that all entries in $\boldsymbol{u}^f$ are retained so that it can be assumed that $N^1_{\boldsymbol{u}} = N^2_{\boldsymbol{u}} = \ldots = N_{\boldsymbol{u}}$. This is in contrast to §3.1.3 where entries in $\boldsymbol{u}^f$ are removed if no boundary

candidate perpendicular to the model contour is sufficiently close. Robust handling of missing boundaries is discussed in §3.2.7.

3.2.5 Joint 2D+t Formulations

Optimising each E^f_{FIT} with respect to $\boldsymbol{u}^f$ and X^f allows the constraints on $\boldsymbol{l}^f$ to be removed from the "track-to-last" algorithm described in §3.1.6, but the sequential and time-dependent nature of the algorithm persists. To achieve reliable segmentation of sequences of arbitrary length and allow user-interaction at any frame it is necessary to abandon the sequential optimisation of (3.5) and instead perform joint minimisation with respect to all $\left\{X^f\right\}_{f=1}^F$, $\left\{\boldsymbol{u}^f\right\}_{f=1}^F$, and $\left\{\boldsymbol{l}^f\right\}_{f=1}^F$. However, (3.5) does not lend itself to joint minimisation because of the presence of $\tilde{l}_i^f$ which is defined in terms of u_i^{f+1} and X^f (3.3).

Consider a straightforward, but incorrect, algorithm for jointly minimising (3.5). Assume that initialisations for all $\left\{X^f\right\}_{f=1}^F$ are available. Next, initialise $\left\{\boldsymbol{u}^f\right\}_{f=1}^F$ so that the model points are uniformly distributed along the length of the model contour in each frame (as in §3.1.3). Given $\left\{X^f\right\}_{f=1}^F$ and $\left\{\boldsymbol{u}^f\right\}_{f=1}^F$, set $\left\{\tilde{\boldsymbol{l}}^f\right\}_{f=1}^{F-1}$ using (3.3) and subsequently solve for $\left\{\boldsymbol{l}^f\right\}_{f=1}^F$ by minimising the sum of E^f_{FIT}, $\mathcal{A}$ and E_{G} terms. Now, it is tempting to minimise E^f_{FIT} and $\mathcal{R}$ with respect to $\left\{\boldsymbol{u}^f\right\}_{f=1}^F$ and $\left\{X^f\right\}_{f=1}^F$ but this would result in model contours which are *not* anatomically consistent. This is because, while $\boldsymbol{\phi}^{f+1}_{l_i^{f+1}}$ has been chosen to have similar appearance to $\boldsymbol{\phi}^f_{\tilde{l}_i^f}$, the minimisation of E^f_{FIT} with respect to X^f is *not* dependent on l_i^{f+1}, $\tilde{l}_i^f$ or u_i^{f+1}; only l_i^f and u_i^f. As a result, after optimising $\left\{\boldsymbol{u}^f\right\}_{f=1}^F$ and $\left\{X^f\right\}_{f=1}^F$ there is no guarantee that $\boldsymbol{\phi}^f_{\tilde{l}_i^f}$ is close to $\chi\left(u_i^{f+1}, X^f\right)$ and the model contours are not anatomically consistent. Importantly, this guarantee is maintained in the "track-to-last" algorithm because X^f and $\boldsymbol{u}^{f+1}$ are fixed when optimising X^{f+1}.

To summarise, the definition of $\tilde{l}_i^f$ and $\mathcal{A}$ terms in (3.5) model the concept that identical points on the model contours in adjacent frames should have similar appearance. However, using $\tilde{l}_i^f$ to achieve this is problematic because its implicit time-dependence prevents joint optimisation of (3.5).

3.2.5.1 Shared Correspondences

Let $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$ define a reduced set of correspondences, where u_i^f is shared between frames f and $f+1$. Also, let $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ denote a set of boundary candidate indices where l_i^f indexes the boundary candidate in frame f corresponding to u_i^f. Furthermore,

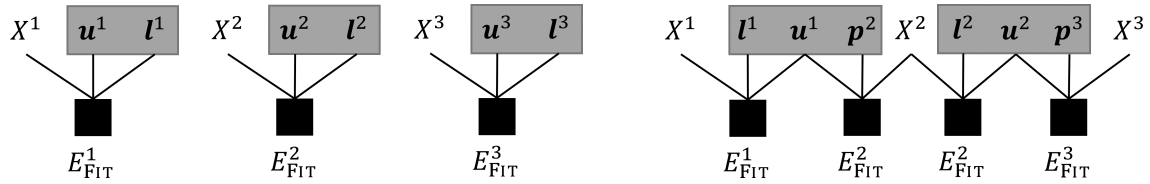


Figure 3.5: Schematics showing the dependencies for E^f_{FIT} ($F = 3$) for (a) $E_{\text{TRACKTOLAST}}$ and (b) E_{SHARED}, with grouped correspondences and boundary candidate indices shown in gray. In E_{SHARED}, correspondences $\boldsymbol{u}^1$ and $\boldsymbol{u}^2$ are used twice so that the unconstrained joint minimum is anatomically consistent (this is not true for $E_{\text{TRACKTOLAST}}$).

let $\left\{\boldsymbol{p}^f\right\}_{f=2}^F$ denote a second set where p_i^{f+1} indexes the boundary candidate in frame $f+1$ corresponding to u_i^f. An alternative formulation to (3.5) is then:

$$\begin{aligned}
&E_{\text{SHARED}}\left(\left\{X^f\right\}_{f=1}^F, \left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}, \left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}, \left\{\boldsymbol{p}^f\right\}_{f=2}^F\right) \\
&\quad = \sum_{f=1}^{F-1} E^f_{\text{FIT}}\left(X^f, \boldsymbol{u}^f, \boldsymbol{l}^f\right) + E^{f+1}_{\text{FIT}}\left(X^{f+1}, \boldsymbol{u}^f, \boldsymbol{p}^{f+1}\right) \\
&\quad + \sum_{f=1}^{F-1}\sum_{i=1}^{N_{\boldsymbol{u}}} \left\{\lambda_{\mathcal{A}}\mathcal{A}\left(\boldsymbol{\vartheta}^f_{l_i^f}, \boldsymbol{\vartheta}^{f+1}_{p_i^{f+1}}\right) + \lambda_{\text{G}} E_{\text{G}}\left(\boldsymbol{\phi}^f_{l_i^f}, \boldsymbol{\phi}^{f+1}_{p_i^{f+1}}\right)\right\} \\
&\quad + \sum_{f=1}^{F} \lambda_{\mathcal{R}}\mathcal{R}\left(X^f\right)
\end{aligned} \tag{3.12}$$

where λ_{G} is a constant, and, in comparison to $E_{\text{TRACKTOLAST}}$, $\left\{\boldsymbol{p}^f\right\}_{f=2}^F$ have been introduced as explicit parameters in place of $\tilde{l}_i^f$ and anatomical consistency is enforced by introducing the additional E^{f+1}_{FIT} term (Figure 3.5).

Now, assume that initialisations for $\left\{X^f\right\}_{f=1}^F$ are available and let $\bar{X}$ denote the matrix of mean model control points. Each $\boldsymbol{u}^f$ is then initialised to $\bar{\boldsymbol{u}}$, the vector of model coordinates which uniformly distributes points across the model contour defined by $\bar{X}$. $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{p}^f\right\}_{f=2}^F$ are solved by unconstrained joint minimisation of the E^f_{FIT}, E^{f+1}_{FIT}, $\mathcal{A}$, and E_{G} terms in (3.12) — $\boldsymbol{l}^1$ is solved with $\boldsymbol{p}^2$, $\boldsymbol{l}^2$ with $\boldsymbol{p}^3$, and so on. Next, given $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{p}^f\right\}_{f=2}^F$, $\left\{X^f\right\}_{f=1}^F$ and $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$ are refined by jointly minimising the E^f_{FIT}, E^{f+1}_{FIT} and $\mathcal{R}$ terms in (3.12). Anatomical consistency is preserved because each $\boldsymbol{u}^f$ is shared between adjacent pairs of frames. This entire procedure is repeated for a fixed number of iterations or until convergence.

To summarise, the reformulation of $E_{\text{TRACKTOLAST}}$ (3.5) to E_{SHARED} (3.12) removes $\tilde{l}_i^f$, introduces a second set of boundary candidate indices $\left\{\boldsymbol{p}^f\right\}_{f=2}^F$, and shares correspondences between frames so that E_{SHARED} is amenable to joint time-independent

optimisation. The resulting 2D+t segmentation algorithm is simply unconstrained joint minimisation of (3.12), alternating between optimisation of the discrete labels and the continuous parameters.

3.2.5.2 Separate Correspondences

In (3.12) the boundary candidates indexed by l_i^f in frame f and p_i^{f+1} in frame $f+1$ are assumed to correspond *exactly* to the same model coordinate u_i^f. Since boundary candidates occupy discrete image positions, it is desirable to allow for small differences in correspondences. Let $\left\{\boldsymbol{v}^f\right\}_{f=2}^F$ denote another set of correspondences to replace $\boldsymbol{u}^f$ in E_{FIT}^{f+1} in (3.12). E_{SEPARATE} is defined as:

$$
\begin{aligned}
&E_{\text{SEPARATE}}\left(\left\{X^f\right\}_{f=1}^F, \left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}, \left\{\boldsymbol{v}^f\right\}_{f=2}^F, \left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}, \left\{\boldsymbol{p}^f\right\}_{f=2}^F\right) \\
&\quad= \sum_{f=1}^{F-1} E_{\text{FIT}}^f\left(X^f, \boldsymbol{u}^f, \boldsymbol{l}^f\right) + E_{\text{FIT}}^{f+1}\left(X^{f+1}, \boldsymbol{v}^{f+1}, \boldsymbol{p}^{f+1}\right) \\
&\quad\quad+ \sum_{f=1}^{F-1} \lambda_{\text{C}} \left\{ \underbrace{\sum_{i=1}^{N_{\boldsymbol{u}}} \left\| \chi\left(u_i^f, X^f\right) - \chi\left(v_i^{f+1}, X^f\right) \right\|^2}_{E_{\text{C}}\left(\boldsymbol{u}^f, \boldsymbol{v}^{f+1}, X^f\right)} + E_{\text{C}}\left(\boldsymbol{u}^f, \boldsymbol{v}^{f+1}, X^{f+1}\right) \right\} \\
&\quad\quad+ \sum_{f=1}^{F-1} \sum_{i=1}^{N_{\boldsymbol{u}}} \left\{ \lambda_{\mathcal{A}} \mathcal{A}\left(\boldsymbol{\vartheta}_{l_i^f}^f, \boldsymbol{\vartheta}_{p_i^{f+1}}^{f+1}\right) + \lambda_{\text{G}} E_{\text{G}}\left(\boldsymbol{\phi}_{l_i^f}^f, \boldsymbol{\phi}_{p_i^{f+1}}^{f+1}\right) \right\} \\
&\quad\quad+ \sum_{f=1}^{F} \lambda_{\mathcal{R}} \mathcal{R}\left(X^f\right)
\end{aligned}
\tag{3.13}
$$

where $E_{\text{C}}\left(\boldsymbol{u}^f, \boldsymbol{v}^{f+1}, X^f\right)$ is a consistency penalty that ensures v_i^{f+1} and u_i^f evaluate to similar positions in frame f. The penalty with respect to frame $f+1$ is also included.

Optimisation of E_{SEPARATE} is similar to E_{SHARED}, with the addition that $\left\{\boldsymbol{v}^f\right\}_{f=2}^F$ is initialised similarly to $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$ and is optimised jointly with $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$ and $\left\{X^f\right\}_{f=1}^F$.

3.2.6 Pairwise Boundary Candidate Constraints

Both E_{SHARED} (3.12) and E_{SEPARATE} (3.13) define energies which recover anatomically consistent contours when minimised. In both, for a given i, l_i^f and p_i^{f+1} are penalised so that the optimal boundary candidates are close to their respective model contour correspondences and have similar appearance. But, for a given f, correlation between l_i^f and $l_{i^+}^f$ (p_i^{f+1} and $p_{i^+}^{f+1}$) is *not* modelled[1]. In practice this is problematic if $\left\{X^f\right\}_{f=1}^F$ is

[1] $i^+ \triangleq (i \mod N) + 1$, so that i^+ remains a one-based index. N is context-specific.

initialised poorly because boundary candidates arising from noise or adjacent structures can be selected even though they do not contribute coherently to the structure of interest (examples are given in §3.4.3). Adding geometric pairwise constraints over $\boldsymbol{l}^f$ and $\boldsymbol{p}^f$ to E_{SHARED} and E_{SEPARATE} models first-order correlations between boundary candidates to improve the coherence of selected boundary candidates:

$$E_{\text{SHARED-P}}\left(\ldots,\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1},\left\{\boldsymbol{p}^f\right\}_{f=2}^{F}\right)=E_{\text{SHARED}}\left(\ldots,\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1},\left\{\boldsymbol{p}^f\right\}_{f=2}^{F}\right)+\sum_{f=1}^{F-1}\lambda_{\text{P}}\left\{\underbrace{\sum_{i=1}^{N_{\boldsymbol{u}}}\left\|\boldsymbol{\phi}_{l_i^f}^f-\boldsymbol{\phi}_{l_{i+}^f}^f\right\|^2}_{E_{\text{P}}^f\left(\boldsymbol{l}^f\right)}+E_{\text{P}}^{f+1}\left(\boldsymbol{p}^{f+1}\right)\right\} \tag{3.14}$$

$$E_{\text{SEPARATE-P}}\left(\ldots,\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1},\left\{\boldsymbol{p}^f\right\}_{f=2}^{F}\right)=E_{\text{SEPARATE}}\left(\ldots,\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1},\left\{\boldsymbol{p}^f\right\}_{f=2}^{F}\right)+\sum_{f=1}^{F-1}\lambda_{\text{P}}\left\{E_{\text{P}}^f\left(\boldsymbol{l}^f\right)+E_{\text{P}}^{f+1}\left(\boldsymbol{p}^{f+1}\right)\right\} \tag{3.15}$$

where E_{P}^f ensures that boundary candidates selected by adjacent model points in frame f are close, and $\ldots$ is notational shorthand to represent all additional (continuous) parameters passed onto E_{SHARED} and E_{SEPARATE}. The definition for E_{P}^f given in (3.14) is for a closed model contour; the upper limit of the summation is $N_{\boldsymbol{u}}-1$ for an open contour.

Importantly, geometric constraints have been proposed previously in [12], but their discrete optimisation algorithm limited their application to open contours only. Overcoming this restriction so that pairwise constraints can be applied to closed contours, as defined in $E_{\text{SHARED-P}}$, is explained in §3.3.4.1.

3.2.7 Robust Boundary Candidate Selection

The "track-to-last" algorithm is robust to missing boundary points because all model coordinates with no boundary candidates perpendicular and sufficiently close to the model contour are removed from $\boldsymbol{u}^f$ (§3.1.3). However, introducing correspondences as free parameters and removing all constraints on boundary candidate selection (§3.2.4) means that *all* entries in $\boldsymbol{u}^f$ are retained and missing boundaries are not explicitly modelled in (3.12), (3.13), (3.14) and (3.15).

The simplest option is to ignore the deficiency in the model. This is sufficient in practice to handle small missing regions because each u_i^f (and v_i^f) is free and if

an incorrect boundary candidate is selected then the correspondence is adjusted to reduce the error (examples are given in §3.4.3). By contrast, if each u_i^f (v_i^f) is fixed — as in the "track-to-last" algorithm — then the model contour is locally stretched or deformed.

Another option is to explicitly model missing boundary candidates by introducing *null* label values which represent "no boundary candidate", extending E_{FIT}^f so that a fixed robust penalty is incurred when a null label value is selected. However, introducing new label values also requires extending the definitions of $\mathcal{A}$ and E_{G}. The complexity induced by both of these extensions is the focus of the remainder of this section.

To simplify discussion, E_{SHARED} (3.12) will be considered in isolation. To start, let ζ_i^f (ξ_i^f) denote the null label value for label l_i^f (p_i^f). E_{FIT}^f is then replaced with $\breve{E}_{\text{FIT}}^f$:

$$\breve{E}_{\text{FIT}}^f\left(X^f, \boldsymbol{u}^f, \boldsymbol{l}^f\right) = \sum_{i=1}^{N_{\boldsymbol{u}}} \begin{cases} \left\| \boldsymbol{\phi}_{l_i^f}^f - \chi\left(u_i^f, X^f\right) \right\|^2, & l_i^f \neq \zeta_i^f \\ \kappa, & l_i^f = \zeta_i^f \end{cases}$$

where κ is the robust penalty. The definition of $\breve{E}_{\text{FIT}}^{f+1}$ with $\boldsymbol{p}^{f+1}$ is similar.

Extending the $\mathcal{A}$ and E_{G} terms is more complex and ad-hoc because it is now necessary to define "position" and "appearance" for each ζ_i^f and ξ_i^f. Assuming that $\left\{X^f\right\}_{f=1}^F$ and $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$ are given, one natural extension is for each null label value to take the position and appearance of its *current* corresponding point on the model contour. This is achieved by extending the available boundary candidates and appearance vectors for each frame.

Let $\breve{\phi}^f$ denote the *extended* boundary candidate matrix for frame f and assume that ζ_i^f and ξ_i^f are valid column indices for $\breve{\phi}^f$. Now, for $1 \leq f \leq F-1$, $\breve{\boldsymbol{\phi}}_{\zeta_i^f}^f \leftarrow \chi\left(u_i^f, X^f\right)$ and $\breve{\boldsymbol{\phi}}_{\xi_i^{f+1}}^{f+1} \leftarrow \chi\left(u_i^f, X^{f+1}\right)$. Also, let $\breve{\vartheta}^f$ denote the extended appearance matrix, where $\breve{\boldsymbol{\vartheta}}_{\zeta_i^f}^f$ takes the appearance vector derived from the image at $\chi\left(u_i^f, X^f\right)$ (a similar definition applies for $\breve{\boldsymbol{\vartheta}}_{\xi_i^{f+1}}^{f+1}$).

With $\breve{\phi}^f$ and $\breve{\vartheta}^f$, $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{p}^f\right\}_{f=1}^{F-1}$ can be found by minimising an energy

almost identical to E_{SHARED} (3.12):

$$\begin{aligned}
&\breve{E}_{\text{SHARED}}\left(\left\{X^f\right\}_{f=1}^{F}, \left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}, \left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}, \left\{\boldsymbol{p}^f\right\}_{f=2}^{F}\right) \\
&\quad = \sum_{f=1}^{F-1} \breve{E}_{\text{FIT}}^{f}\left(X^f, \boldsymbol{u}^f, \boldsymbol{l}^f\right) + \breve{E}_{\text{FIT}}^{f+1}\left(X^{f+1}, \boldsymbol{u}^f, \boldsymbol{p}^{f+1}\right) \\
&\quad + \sum_{f=1}^{F-1} \sum_{i=1}^{N_{\boldsymbol{u}}} \lambda_{\mathcal{A}} \mathcal{A}\left(\breve{\boldsymbol{\vartheta}}_{l_i^f}^{f}, \breve{\boldsymbol{\vartheta}}_{p_i^{f+1}}^{f+1}\right) + \lambda_{\text{G}} E_{\text{G}}\left(\breve{\boldsymbol{\phi}}_{l_i^f}^{f}, \breve{\boldsymbol{\phi}}_{p_i^{f+1}}^{f+1}\right) \\
&\quad + \sum_{f=1}^{F} \lambda_{\mathcal{R}} \mathcal{R}\left(X^f\right)
\end{aligned}$$

Given $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{p}^f\right\}_{f=1}^{F-1}$ it is then necessary to refine $\left\{X^f\right\}_{f=1}^{F}$ and $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$. Initially, this seems as straightforward as minimising the $\breve{E}_{\text{FIT}}^{f}$ and $\mathcal{R}$ terms, updating only those entries in $\boldsymbol{u}^f$ that have labels not equal to their null label values. While this algorithm is intuitive and results in anatomically consistent model contours, it is strictly not the exact minimum because of the dependence of $\breve{\vartheta}^f$ and $\breve{\phi}^f$ on $\left\{X^f\right\}_{f=1}^{F}$ and $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$. Including the E_{G} terms in the minimisation is straightforward — each is just a geometric penalty term. However, the $\mathcal{A}$ terms are more problematic because continuous optimisation directly over the image is required. Due to this complexity, robust modelling of missing boundaries was not investigated past an initial implementation and is not considered in the sections or experiments that follow in this chapter.

3.3 Framework Components

Up until now, definitions for the framework components (χ, ν, $\mathcal{R}$, $\mathcal{A}$) and details of the optimisation algorithms have been omitted. This was done to make the distinctions between the 2D/2D+t segmentation models clear and independent of component and algorithm choices. The purpose of this section is to give specific definitions for the framework components used in the experiments that follow, and describe the necessary discrete and continuous optimisation algorithms.

3.3.1 Model Contour

A closed uniform quadratic B-spline was used for the model contour [18, p. 41]. With $X \in \mathbb{R}^{2\times N_X}$ denoting the N_X control points, the domain of the model contour is the half-open interval $\Omega = [0, N_X) \subset \mathbb{R}$. Given a model coordinate u, let $i = \lfloor u \rfloor$ denote

the (zero-based) segment index and let $t = u - \lfloor u \rfloor$ denote the segment coordinate. The position on the model contour is then given by:

$$\chi(u, X) = \left(\frac{1}{2} - t + \frac{1}{2}t^2\right) \boldsymbol{x}_{i+1} + \left(\frac{1}{2} + t - t^2\right) \boldsymbol{x}_{i\oplus 1+1} + \left(\frac{1}{2}t^2\right) \boldsymbol{x}_{i\oplus 2+1} \tag{3.16}$$

where $i \oplus j \triangleq (i+j) \mod N$, where N is context-specific. Uniform quadratic[2] B-splines are appropriate because they naturally model smooth curves, are twice differentiable, and each contour point is only dependent on a limited number of the control points which enables accurate modelling of local deformations. ν is defined as the normalised tangent vector function rotated[3] by $90\,^\circ$.

3.3.2 Regulariser

For demonstrative purposes, a very simple regulariser for the closed model contour was used which encourages adjacent control points to be as close as possible to each other:

$$\mathcal{R}(X) = \sum_{i=1}^{N_X} \|\boldsymbol{x}_i - \boldsymbol{x}_{i^+}\|^2 \tag{3.17}$$

(The upper limit of the summation is $N_X - 1$ for an open model contour.)

Physically motivated regularisers, such as minimum length and minimum curvature, are also suitable and simply expressed for model contour definitions which are linear in X [78].

3.3.3 Appearance Similarity

In the experiments that follow, zero-mean Gaussian noise was used to generate colour profiles in the synthetic sequences used for 2D+t segmentation assessment (§3.4.2). Therefore, SSD was used to measure appearance similarity:

$$\mathcal{A}(\boldsymbol{\vartheta}_i, \boldsymbol{\vartheta}_j) = \frac{1}{\mathcal{W}} \left\|\boldsymbol{\vartheta}_i - \boldsymbol{\vartheta}_j\right\|^2 \tag{3.18}$$

where $\mathcal{W}$ is the number of elements in $\boldsymbol{\vartheta}_i$ and $\boldsymbol{\vartheta}_j$.

[2]The choice of the degree of the B-spline is relatively arbitrary and similar results can be expected with cubic or quartic B-splines.

[3]The direction of rotation depends on whether inward- or outward-facing normals are required, and whether contour control points are labelled clockwise or anti-clockwise.

3.3.4 Optimisation

E_{SHARED} (3.12), E_{SEPARATE} (3.13), $E_{\text{SHARED-P}}$ (3.14), and $E_{\text{SEPARATE-P}}$ (3.15) are all completely defined given the definitions for χ (3.16), $\mathcal{R}$ (3.17) and $\mathcal{A}$ (3.18). The algorithms for performing approximate and exact (where possible) optimisation of each energy are described now.

3.3.4.1 Discrete Optimisation

***Without* Pairwise Boundary Candidate Constraints** Given values for the continuous variables — $\left\{X^f\right\}_{f=1}^{F}$, $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$, and (potentially) $\left\{\boldsymbol{v}^f\right\}_{f=2}^{F}$ — it is necessary to solve $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{p}^f\right\}_{f=2}^{F}$. To start, consider E_{SHARED} with $F = 2$ and discrete unknowns $\boldsymbol{l}^1$ and $\boldsymbol{p}^2$. Minimisation of E_{SHARED} is then a discrete minimisation problem over the augmented vector $\breve{\boldsymbol{l}} = [\boldsymbol{l}^1 \mid \boldsymbol{p}^2]$ of length $2N_{\boldsymbol{u}}$ where:

1. For $1 \leq i \leq N_{\boldsymbol{u}}$, $\breve{l}_i$ is the index of the boundary candidate in frame 1 corresponding to u_i^1, i.e. $\breve{l}_i \equiv l_i^1$.

2. For $N_{\boldsymbol{u}} + 1 \leq i \leq 2N_{\boldsymbol{u}}$, $\breve{l}_i$ is the index of the boundary candidate in frame 2 corresponding to $u_{i-N_{\boldsymbol{u}}}^1$, i.e. $\breve{l}_i \equiv p_{i-N_{\boldsymbol{u}}}^2$.

Additionally, the *label space* of $\breve{\boldsymbol{l}}$ is the vector of boundary candidate indices for *both* frames and is of size $L = N_\phi^1 + N_\phi^2$.

With reference to (3.12), let $\theta_{\text{UNARY}} \in \mathbb{R}^{2N_{\boldsymbol{u}} \times L}$ denote the matrix of *unary potentials*, where $(\theta_{\text{UNARY}})_{i,j}$ is the fitting error of $\breve{l}_i$ taking label j. For $1 \leq i \leq N_{\boldsymbol{u}}$, it is given by the i^{th} summand of E_{FIT}^1; for $N_{\boldsymbol{u}} + 1 \leq i \leq 2N_{\boldsymbol{u}}$ it is summand $i - N_{\boldsymbol{u}}$ of E_{FIT}^2. Importantly, θ_{UNARY} is block-sparse because the first (second) $N_{\boldsymbol{u}}$ elements of $\breve{\boldsymbol{l}}$ can only take labels corresponding to boundary candidate indices in frame 1 (2).

In (3.12), the $\mathcal{A}$ and E_{G} terms each take l_i^1 ($\breve{l}_i$) and p_i^2 ($\breve{l}_{i+N_{\boldsymbol{u}}}$) as arguments. That is, $\breve{l}_i$ and $\breve{l}_{i+N_{\boldsymbol{u}}}$ are *adjacent*. Now, the matrix of *pairwise potentials* is denoted by $\theta_{\text{PAIRWISE}} \in \mathbb{R}^{L \times L}$, where $(\theta_{\text{PAIRWISE}})_{i,j}$ is the error of adjacent elements in $\breve{\boldsymbol{l}}$ taking labels i and j. For (3.12), θ_{PAIRWISE} is sparse and symmetric. Sparsity arises because there is no error term which is defined for labels denoting boundary candidate indices in the same frame; symmetry follows from the definitions of $\mathcal{A}$ and E_{G}.

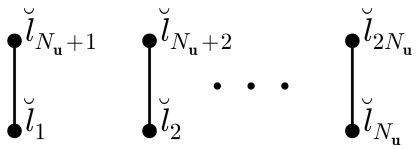


Figure 3.6: $N_{\boldsymbol{u}}$ independent 2-node graphical models. Filled circles represent nodes and solid lines represent edges.

With θ_{UNARY} and θ_{PAIRWISE}, minimising E_{SHARED} (for $F = 2$) with respect to $\boldsymbol{l}^1$ and $\boldsymbol{p}^2$ is equivalent to finding $\breve{\boldsymbol{l}}$ to minimise:

$$\begin{aligned} E\left(\breve{\boldsymbol{l}}\right) &= \sum_{i=1}^{2N_{\boldsymbol{u}}} \left(\theta_{\text{UNARY}}\right)_{i,\breve{l}_i} + \sum_{i=1}^{N_{\boldsymbol{u}}} \left(\theta_{\text{PAIRWISE}}\right)_{\breve{l}_i,\breve{l}_{i+N_{\boldsymbol{u}}}} \\ &= \sum_{i=1}^{N_{\boldsymbol{u}}} \left\{ \left(\theta_{\text{UNARY}}\right)_{i,\breve{l}_i} + \left(\theta_{\text{UNARY}}\right)_{i+N_{\boldsymbol{u}},\breve{l}_{i+N_{\boldsymbol{u}}}} + \left(\theta_{\text{PAIRWISE}}\right)_{\breve{l}_i,\breve{l}_{i+N_{\boldsymbol{u}}}} \right\} \end{aligned}$$

which can be performed independently over each summand. Minimisation of each summand with respect to $\left(\breve{l}_i, \breve{l}_{i+N_{\boldsymbol{u}}}\right)$ is done by enumerating all *feasible* $L \times L$ joint solutions and taking the minimum. Importantly, this is identical to the Minimum Sum (`MIN-SUM`) algorithm (`max-sum` in [16, p. 411]) applied to $N_{\boldsymbol{u}}$ independent 2-node graphical models (Figure 3.6).

For $F > 2$, discrete optimisation of E_{SHARED} is straightforward because each pair $\left(\boldsymbol{l}^f, \boldsymbol{p}^{f+1}\right)$ can be solved independently using the above algorithm. This separability follows from the independence of each summand in (3.12). Given all continuous variables, E_{SEPARATE} also has identical form to E_{SHARED} and can be optimised by the same algorithm. Finally, unconstrained discrete optimisation for single-frame model contour fitting (3.2) is trivial. In the absence of θ_{PAIRWISE}, each element of $\breve{\boldsymbol{l}}$ is set to the index of the minimum element of each row of θ_{UNARY}.

***With* Pairwise Boundary Candidate Constraints** To start, consider (3.2) with the addition of pairwise boundary candidate constraints from §3.2.6:

$$E\left(\boldsymbol{l} \mid \boldsymbol{u}, X\right) = \underbrace{\sum_{i=1}^{N_{\boldsymbol{u}}} \left\| \boldsymbol{\phi}_{l_i} - \chi\left(u_i, X\right) \right\|^2}_{E_{\text{FIT}}(X,\boldsymbol{u},\boldsymbol{l})} + \lambda_{\text{P}} \underbrace{\sum_{i=1}^{N_{\boldsymbol{u}}} \left\| \boldsymbol{\phi}_{l_i} - \boldsymbol{\phi}_{l_{i^+}} \right\|^2}_{E_{\text{P}}(\boldsymbol{l})} + \lambda_{\mathcal{R}} \mathcal{R}\left(X\right) \qquad (3.19)$$

where f superscripts have been omitted for convenience. Here, $\breve{\boldsymbol{l}} = \boldsymbol{l}$ and the label space is the vector of boundary candidate indices and is of size $L = N_{\phi}$. As before, θ_{UNARY} denotes the matrix of unary potentials, where $\left(\theta_{\text{UNARY}}\right)_{i,j}$ is the weighted squared distance of the model point at coordinate u_i to the boundary candidate $\boldsymbol{\phi}_j$. With

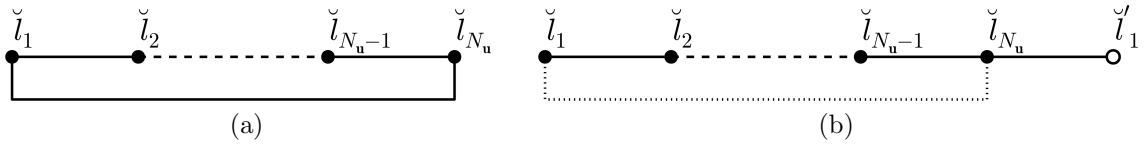


Figure 3.7: (a) Equivalent graphical model for (3.20). Dashed lines represent repeated nodes and edges. (b) Graphical model used in the CSP algorithm with the cycle removed (dotted edge) and the phantom node $\breve{l}'_1$ added (hollow circle).

E_P, each element of $\breve{\boldsymbol{l}}$ *cannot* be solved independently because $\breve{l}_i$ and $\breve{l}_{i^+}$ are adjacent. Now, θ_PAIRWISE denotes the matrix of pairwise potentials, where $(\theta_\text{PAIRWISE})_{i,j}$ is equal to the weighted squared distance between boundary candidates $\boldsymbol{\phi}_i$ and $\boldsymbol{\phi}_j$. With respect to $\breve{\boldsymbol{l}}$, an equivalent energy to (3.19) is:

$$E\left(\breve{\boldsymbol{l}}\right) = \sum_{i=1}^{N_{\boldsymbol{u}}} \left\{ (\theta_\text{UNARY})_{i,\breve{l}_i} + (\theta_\text{PAIRWISE})_{\breve{l}_i,\breve{l}_{i^+}} \right\} \tag{3.20}$$

which is equivalent to the graphical model in Figure 3.7a. This graphical model contains a cycle and cannot be solved directly with standard MIN-SUM. Exact minimisation is possible using the branch-and-bound Circular Shortest Path (CSP) algorithm by Appleton and Sun [6].

In the CSP algorithm, a *phantom* node with unary potentials of zero is added to the model and the edge joining the (arbitrary) start and end nodes is moved between the end and phantom nodes (Figure 3.7b). The algorithm proceeds by solving the surrogate model multiple times using MIN-SUM, each time further restricting the label space for the start and phantom nodes of the current minimum configuration. The algorithm terminates when the current minimum solution for the surrogate model is a valid solution to the original model — it has identical labels for the phantom and start nodes. This is the global minimum solution for the original model because further restriction on the label space of the start and phantom nodes cannot give rise to a lower energy solution [6].

Now consider $E_\text{SHARED-P}$ with $F = 2$. Like E_SHARED, $\breve{\boldsymbol{l}} = [\boldsymbol{l}^1 \mid \boldsymbol{p}^2]$, $L = N_\phi^1 + N_\phi^2$, and θ_UNARY is defined identically. However, θ_PAIRWISE is no longer sparse because the pairwise boundary candidate constraints are error terms for boundary candidates in

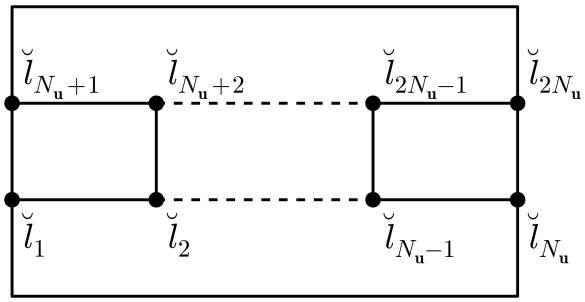


Figure 3.8: Equivalent graphical model for (3.21).

the same frame. The equivalent energy is given by:

$$E\left(\breve{\boldsymbol{l}}\right) = \sum_{i=1}^{2N_{\boldsymbol{u}}} \left(\theta_{\text{UNARY}}\right)_{i,\breve{l}_i} + \sum_{i=1}^{N_{\boldsymbol{u}}} \left(\theta_{\text{PAIRWISE}}\right)_{\breve{l}_i,\breve{l}_{i+N_{\boldsymbol{u}}}} + \sum_{i=1}^{N_{\boldsymbol{u}}} \left(\theta_{\text{PAIRWISE}}\right)_{\breve{l}_i,\breve{l}_{i^+}} + \sum_{i=1}^{N_{\boldsymbol{u}}} \left(\theta_{\text{PAIRWISE}}\right)_{\breve{l}_{i+N_{\boldsymbol{u}}},\breve{l}_{(i+N_{\boldsymbol{u}})^+}} \tag{3.21}$$

and is a first-order Markov Random Field (MRF) [23] with the corresponding graphical model shown in Figure 3.8. Similar to (3.20), (3.21) cannot be solved with MIN-SUM. However, no bespoke algorithm akin to CSP exists to globally minimise (3.21).

An approximate solution is obtained as follows. First, Prim's algorithm [39, pp. 634–636] is used to generate a random spanning tree from the graph implied by the pairwise terms in (3.21). Specifically, edges are assigned random weights from the unit uniform distribution and the root node is chosen at random. An *approximation* to (3.21) is then constructed by retaining only the pairwise terms which are present as edges in the random spanning tree. This is then minimised using MIN-SUM to provide an initial value for $\breve{\boldsymbol{l}}$.

The initial value for $\breve{\boldsymbol{l}}$ is the *global* minimum of the *approximate* energy; it is not the global minimum of (3.21). The labelling is subsequently refined using Quadratic Pseudo-Boolean Optimisation (QPBO) [64, 80] to minimise (3.21). Specifically, for every label α in the label space, a binary expansion move [23] is solved using QPBO to determine whether each element in $\breve{\boldsymbol{l}}$ not equal to α should change to α (or not). QPBO is particularly useful because it is applicable to *non-submodular* problems, although it can only guarantee a partial solution for such problems. Importantly, the α-expansion move subproblems are non-submodular by definition of the pairwise terms from $E_{\text{SHARED-P}}$. To show this, a multivalued variable problem is submodular if the pairwise

terms satisfy:

$$(\theta_{\text{PAIRWISE}})_{\beta,\theta} + (\theta_{\text{PAIRWISE}})_{\alpha,\alpha} \leq (\theta_{\text{PAIRWISE}})_{\beta,\alpha} + (\theta_{\text{PAIRWISE}})_{\alpha,\theta}$$

for all labels α, β, θ [80]. $(\theta_{\text{PAIRWISE}})_{\alpha,\alpha} = 0$ as a result of the definitions of E_{G}, $\mathcal{A}$, and E_{P} so that the submodularity condition reduces to:

$$(\theta_{\text{PAIRWISE}})_{\beta,\theta} \leq (\theta_{\text{PAIRWISE}})_{\beta,\alpha} + (\theta_{\text{PAIRWISE}})_{\alpha,\theta}$$

which is the triangle inequality. Considering just E_{P} in isolation, this condition is not fulfilled because squared distance does not obey the triangle inequality; the problem is non-submodular.

Importantly, the above algorithm is approximate and non-deterministic. In practice it is run multiple times to generate different spanning trees and the solution with the lowest energy is chosen.

Finally, discrete optimisation of $E_{\text{SHARED-P}}$ and $E_{\text{SEPARATE-P}}$ for $F > 2$ is straightforward as both problems are equivalent and each pair $\left(\boldsymbol{l}^f, \boldsymbol{p}^{f+1}\right)$ can be solved independently.

3.3.4.2 Continuous Optimisation

With values for the discrete variables — $\left\{\boldsymbol{l}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{p}^f\right\}_{f=2}^{F}$ — subsequent refinement of the continuous variables — $\left\{X^f\right\}_{f=1}^{F}$, $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$, and potentially $\left\{\boldsymbol{v}^f\right\}_{f=2}^{F}$ — is required. To start, consider local *joint* continuous optimisation of E_{SHARED} (3.12) with respect to $\left\{X^f\right\}_{f=1}^{F}$ and $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$. Let $\boldsymbol{z}$ denote the vector of all concatenated continuous variables. Since E_{FIT}^{f}, E_{FIT}^{f+1} (3.26), and $\mathcal{R}$ (3.17) are all functions that are sums of squared residuals, E_{SHARED} in its most general form is:

$$E_{\text{SHARED}}(\boldsymbol{z}) = \frac{1}{2}\sum_i \left(f_i(\boldsymbol{z})\right)^2 + k \tag{3.22}$$

where f_i are the (potentially non-linear) residual functions and k is the contribution from terms which are functions of the discrete parameters only. The dependence on discrete parameters is omitted for notational convenience. Local continuous optimisation of (3.22) is straightforward using the Levenberg-Marquardt (`LM`) algorithm.

The Levenberg-Marquardt (`LM`) Algorithm Let $\delta\boldsymbol{z}$ denote a small change in $\boldsymbol{z}$. Using a first-order Taylor expansion, the energy evaluated at $\boldsymbol{z} + \delta\boldsymbol{z}$ is approximately:

$$E_{\text{SHARED}}(\boldsymbol{z} + \delta\boldsymbol{z}) \approx \underbrace{\frac{1}{2}\sum_i \left(f_i(\boldsymbol{z}) + \boldsymbol{\partial}_i{}^{\top}\delta\boldsymbol{z}\right)^2}_{E'_{\text{SHARED}}(\delta\boldsymbol{z})} + k$$

where $\boldsymbol{\partial}_i$ is the vector of derivatives of f_i evaluated at $\boldsymbol{z}$. In vector form:

$$E'_{\text{SHARED}}(\delta\boldsymbol{z}) = \frac{1}{2}\left\|\boldsymbol{f} + J\delta\boldsymbol{z}\right\|^2$$

where $\boldsymbol{f}$ is the vector of residuals evaluated at $\boldsymbol{z}$, and J is the Jacobian matrix where the i^{th} row is equal to $\boldsymbol{\partial}_i{}^\top$. Taking the derivative of E'_{SHARED} and setting all components to zero:

$$\begin{aligned}\frac{\partial E'_{\text{SHARED}}}{\partial\delta\boldsymbol{z}}(\delta\boldsymbol{z}) &= J^\top\left(\boldsymbol{f} + J\delta\boldsymbol{z}\right) = \mathbf{0}\\ \Rightarrow \left(J^\top J\right)\delta\boldsymbol{z} &= -J^\top\boldsymbol{f}\end{aligned} \tag{3.23}$$

where $-J^\top\boldsymbol{f}$ is the negative gradient. In the Gauss-Newton algorithm, (3.23) is solved directly — provided J is full-rank — and $\delta\boldsymbol{z}$ is used as a *direction* to find an update for $\boldsymbol{z}$. That is:

$$\boldsymbol{z} \leftarrow \boldsymbol{z} + \alpha\delta\boldsymbol{z}$$

where α is found by *line search* [89].

In Levenberg's algorithm, a *damped* system is considered instead [99]:

$$\left(J^\top J + \frac{1}{\mu}I\right)\delta\boldsymbol{z} = -J^\top\boldsymbol{f} \tag{3.24}$$

where I is the identity matrix and μ is a scalar parameter such that the coefficient matrix of $\delta\boldsymbol{z}$ is positive definite. Solving (3.24), $\delta\boldsymbol{z}$ is then added directly to $\boldsymbol{z}$. If the proposed update to $\boldsymbol{z}$ decreases the function energy, it is accepted and μ is increased. Otherwise, μ is decreased[4] and (3.24) is solved again. These steps are repeated until a convergence criteria (e.g. $\mu < \mu^0$) is fulfilled.

An alternative formulation to (3.24) replaces I with the square root of the diagonal of $J^\top J$ [99]. This is sometimes preferred because (3.24) is *not* scale-invariant. Also, faster truncated Newton methods are available which approximately solve (3.24) [114]. Here, `LM` refers to Levenberg's algorithm as given in (3.24) and [96].

Local Continuous Minimisation of E_{Shared} using `LM` Application of `LM` has two requirements: evaluation of the system Jacobian and a suitable procedure for applying updates $\delta\boldsymbol{z}$ to $\boldsymbol{z}$.

Calculation of J for E_{SHARED} is straightforward. The derivatives of $\mathcal{R}$ with respect to X^f are well-defined, and the derivatives of E^f_{FIT} with respect to u^f_i and X^f are evaluable using the chain-rule and derivatives of χ and ν.

[4]The exact functions for increasing and decreasing μ vary between implementations [96].

Defining a suitable procedure for updating $\boldsymbol{z}$ with $\delta\boldsymbol{z}$ is slightly more subtle. By default, `LM` performs Euclidean vector addition: $\boldsymbol{z} + \delta\boldsymbol{z}$. With reference to §3.3.1, this is acceptable for $\left\{X^f\right\}_{f=1}^{F}$ since $\boldsymbol{x}_i^f \in \mathbb{R}^2$. However, the domain of each correspondence is a subset of the real numbers: $u_i^f \in \Omega \subset \mathbb{R}$. To maintain $u_i^f \in \Omega$, "correspondence addition" — denoted by $\boxplus$ — is defined as:

$$u_i^f \boxplus \delta u_i^f = \text{fmod}\left(u_i^f + \delta u_i^f, N_X\right)$$

where fmod is the non-negative modulo function extended to $\mathbb{R}$ given by:

$$\text{fmod}\,(x, y) = x - y\left\lfloor\frac{x}{y}\right\rfloor \tag{3.25}$$

with $\lfloor\cdot\rfloor$ rounding to the closest integer in the direction of $-\infty$ (*not* towards 0). This definition ensures that updated correspondences correctly transition between the start and end segments of the (closed) model contour.

Finally, minimisation of E_{SEPARATE} is almost identical to E_{SHARED} — additional E_{C} terms are included and $\boxplus$ is used to update $\left\{\boldsymbol{v}^f\right\}_{f=2}^{F}$.

A Note on Complexity Except for extremely small problems, the step which dominants the execution time of `LM` is solving the linear system in (3.24).

The size of the square coefficient matrix is equal to the number of parameters in $\boldsymbol{z}$, which includes the model contour control points and *all* model correspondences. Handled naïvely, increasing the number of model correspondences and/or the number of frames increases the computation time of $\delta\boldsymbol{z}$ dramatically, limiting the practicality of jointly optimising $\left\{X^f\right\}_{f=1}^{F}$, $\left\{\boldsymbol{u}^f\right\}_{f=1}^{F-1}$ and $\left\{\boldsymbol{v}^f\right\}_{f=2}^{F}$. However, due to the near complete independence of the model correspondences[5], $J^\top J$ is sparse and has large block-diagonal sub-blocks so that solving time increases linearly with respect to the number of model correspondences [150]. Further implementation details are explained in §3.4.1.

3.3.5 Oriented Contour Fit

Up until now the fit of the model contour to the selected boundary candidates has been defined in terms of displacement only (§3.1.3). A simple addition is to also consider the difference in orientation between the model contour and selected boundary candidates. This has the practical benefit of discriminating between "interior" and

[5]The inclusion of E_{C} over $\boldsymbol{u}^f$ and $\boldsymbol{v}^{f+1}$ in E_{SEPARATE} breaks total independence.

"exterior" edge responses during boundary candidate selection. The simplest extension to E^f_{FIT} is given by:

$$E^f_{\text{FIT}}\left(X^f, \boldsymbol{u}^f, \boldsymbol{l}^f\right) = \sum_{i=1}^{N^f_{\boldsymbol{u}}} \left\{ \left\| \boldsymbol{\phi}^f_{l^f_i} - \chi\left(u^f_i, X^f\right) \right\|^2 + \lambda_\eta \left\| \boldsymbol{\eta}^f_{l^f_i} - \nu\left(u^f_i, X^f\right) \right\|^2 \right\} \quad (3.26)$$

where η^f is the matrix of N^f_ϕ boundary candidate normals in frame f and λ_η controls its weight. Other measures of orientation dissimilarity such as:

$$\lambda_\eta \left(\arccos\left(\boldsymbol{\eta}^f_{l^f_i} \cdot \nu\left(u^f_i, X^f\right) \right) \right)^2$$

are also suitable but were not investigated further. The definition of E^f_{FIT} in (3.26) is used for boundary candidate selection and model contour fitting in the experiments that follow.

3.4 Experiments

The purpose of this section is to demonstrate the advantages of the joint 2D+t model-based segmentation algorithms described in §3.2.5.1, §3.2.5.2, and §3.2.6 over the "track-to-last" algorithm described in §3.1.5.

First, the influence of free correspondences, unconstrained boundary candidate selection (§3.2.4), and pairwise boundary candidate constraints (§3.2.6) towards single-frame segmentation accuracy and robustness to initialisation are examined qualitatively and quantitatively. The segmentation task considered is long-axis segmentation of the endocardium from 2D echocardiography images, and both synthetic and acquired frames are used for validation. Second, the algorithms of §3.2.6 are compared against the "track-to-last" algorithm of §3.1.5 on the task of 2D+t long-axis segmentation. Synthetic edge maps are utilised to enable quantitative assessment of anatomical consistency. Results on acquired images are also provided to enable qualitative assessment.

3.4.1 Implementation

All solver components were implemented in a combination of C, C++ and Python 2.7 with Numpy 1.7[6] and Scipy 0.13[7] extensions.

[6] `http://www.numpy.org`
[7] `http://www.scipy.org`

For the discrete optimisation, `MIN-SUM` was implemented independently in C++. For `QPBO`, the C++ implementation by Kolmogorov[8] was used, with the modification that α-expansion is only performed on nodes for which α is a valid label.

For the continuous optimisation, the C++ non-linear least squares library `CERES` [2] was used which internally uses `SuiteSparse` [5, 33] to efficiently solve the `LM` linear system (3.24). Importantly, the initial available version of `CERES` only supported problems with fixed Jacobian sparsity patterns. For the models introduced in this chapter, the system Jacobian has a *dynamic* sparsity pattern. This arises as a result of the local support of χ and ν which are used in E_{FIT}. Specifically, as u_i^f and v_i^f change, the *active* model contour control points in χ and ν — which have non-zero Jacobian entries — change too. An extension to `CERES` was written[9] to handle these dynamic sparsity patterns.

The experiments performed in this section were run on an Intel Core i7-4702MQ 2.20GHz processor. All solvers were executed single-threaded.

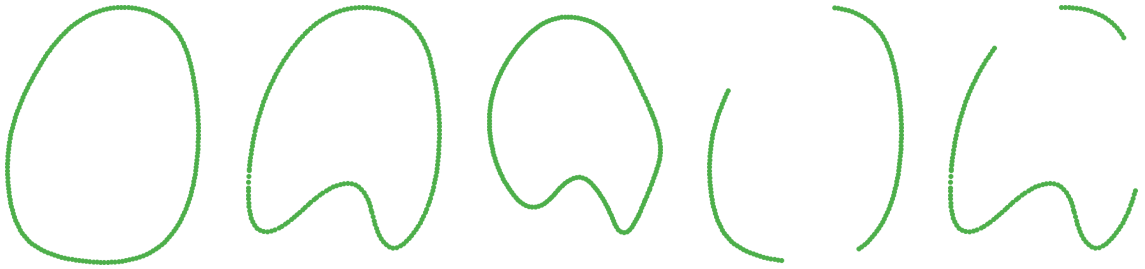
3.4.2 Materials

Synthetic Edge Maps Synthetic edge maps were generated to enable quantitative assessment of segmentation accuracy between the different segmentation algorithms. The advantages of synthetic edge maps are that they contain no spurious edge responses and that the resulting segmentation assessment is isolated from the choice of edge detection algorithm. This is appropriate because the algorithms under scrutiny differ only in how boundary candidates are selected and how the model contour is fitted.

For single-frame segmentation, synthetic edge maps were generated by uniformly sampling $N_\phi = 200$ points on interactively defined uniform quadratic B-spline contours confined within the unit square. Three contours of increasing shape and model complexity were used: `CONVEX-8`, `CONCAVE-8`, and `CONCAVE-12` (Figures 3.9a, 3.9b, and 3.9c). Sets of boundary candidates with missing boundary sections were also generated: `CONVEX-8-MB` and `CONCAVE-8-MB` (Figures 3.9d and 3.9e).

For 2D+t segmentation, synthetic sequences were generated by a similar process. First, initial contours were interactively defined (Figures 3.10a and 3.11a). Next, each contour was deformed by a linear transformation and random perturbations to generate a sequence of contours. Let $\boldsymbol{r}$ denote the vector of F values linearly spaced between 0 and 1 which have been raised to the power of p. Also, let Y^1 denote the

[8] `http://pub.ist.ac.at/~vnk/software.html`
[9] `http://ceres-solver.org/version_history.html`



(a) CONVEX-8 (b) CONCAVE-8 (c) CONCAVE-12 (d) CONVEX-8-MB (e) CONCAVE-8-MB

Figure 3.9: Single-frame segmentation synthetic edge maps. Boundary candidates are green.

matrix of control points defining the initial contour and Y^f denote the control points in frame f. For $f > 1$, each $\boldsymbol{y}_i^f$ was initialised as:

$$\boldsymbol{y}_i^f = s^{r_f} \mathcal{T}\left(r_f \theta\right) \left(\boldsymbol{y}_i^1 - \bar{\boldsymbol{y}}^1\right) + \bar{\boldsymbol{y}}^1 \tag{3.27}$$

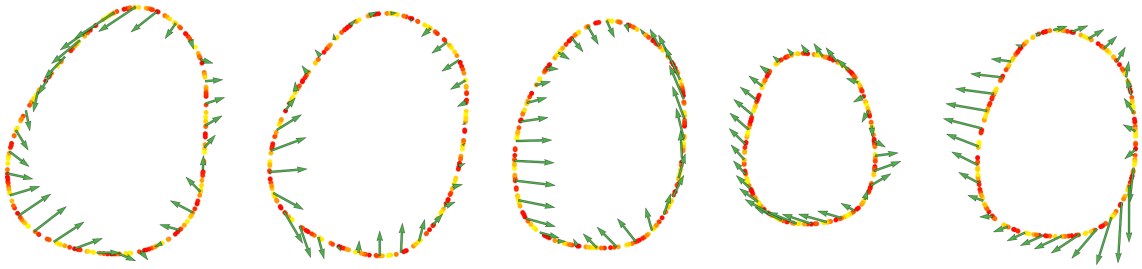
where $\bar{\boldsymbol{y}}^1$ is the centroid of Y^1, s is the maximum scale, θ is the maximum rotation about the centroid, and $\mathcal{T}$ returns a rotation matrix given an input angle. Values of $F = 12$, $p = 1.5$, $s = 0.7$ and $\theta = 20\,^\circ$ were used.

Following (3.27), each sequence was extended to $2F - 1 = 23$ frames by setting $Y^{F+1} = Y^{F-1}, Y^{F+2} = Y^{F-2}, \ldots, Y^{2F-1} = Y^1$. Displacements drawn from an isotropic zero-mean Gaussian with standard deviation 0.01 were added to each $\boldsymbol{y}_i^f$ to simulate (minor) random deformations. Finally, $N_\phi = 200$ boundary candidates were sampled from the model contours defined in each frame. Random perturbations drawn from a zero-mean Gaussian with standard deviation 0.05 were added to each sampling coordinate to ensure that identical model coordinates were *not* sampled in the sequence.

To simulate appearance, 1D colours were assigned to positions along the model contour. The *colour profile* was a cosine function of frequency $f = 20$ defined over normalised arc-length. $32 \times f$ samples were evaluated and zero-mean Gaussian noise with standard deviation 0.05 was added to each. 1D colours were assigned to each boundary candidate by converting the model coordinate to a normalised arc-length position and linearly interpolating between the corresponding colour profile samples.

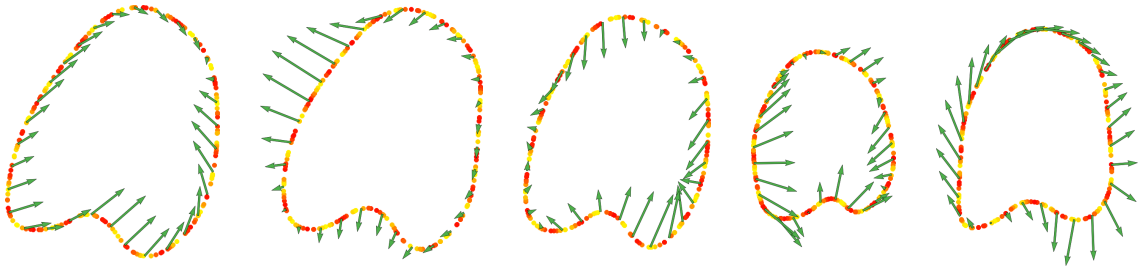
The two generated sequences — CONVEX-8 and CONCAVE-10 — are shown in Figures 3.10 and 3.11 respectively.

2D Echocardiography Slices The 2D echocardiography frames were four-chamber slices extracted from 3D echocardiography acquisitions. The entire dataset consisted of 10 healthy subjects with 11 frames from end-diastole to end-systole. The images were



(a) $f = 1$ (b) $f = 2$ (c) $f = 5$ (d) $f = 12$ (e) $f = 17$

Figure 3.10: Frames from the 2D+t synthetic sequence `CONVEX-8`. The 1D colour of each boundary candidate is shown with false colour between yellow and red. The magnitude and direction of frame-to-frame deformations are shown in green, with the magnitudes scaled by a factor of four for clarity.



(a) $f = 1$ (b) $f = 2$ (c) $f = 5$ (d) $f = 12$ (e) $f = 17$

Figure 3.11: Frames from the 2D+t synthetic sequence `CONCAVE-10`.

acquired from a Philips iE33 ultrasound system with a mean image spatial resolution of 0.85mm × 0.86mm × 0.77mm, standard dimensions of 224 × 208 × 208, and the intensities normalised to the unit interval. Boundary candidates were detected in each frame by convolving each slice with a 2D log-Gabor bandpass filter, computing the feature asymmetry [130], followed by non-maximum suppression [26]. A centre frequency of $f_0 = 0.05\text{mm}^{-1}$ and bandwidth of $b = 2$ octaves were used for the log-Gabor filter. A threshold of twice the geometric mean ($T = 2$) of the local energy was used to remove small edge responses [81].

3.4.3 Single-Frame Segmentation

Four segmentation algorithms were compared to assess the impact of free correspondences, unconstrained boundary candidate selection, and pairwise boundary candidate constraints:

1. `PERPENDICULAR-FIXED`: Single-frame fit (§3.1.3) with E^f_{FIT} from (3.26), fixed correspondences, and perpendicular boundary candidate selection (3.1).

Scale	$\{0.8, 1.0, 1.25\}$
Translation	$\{(x, y) : x \in \{-0.2, 0, 0.2\}, y \in \{-0.2, 0, 0.2\}\}$
Rotation	$\{-20\,^\circ, 0\,^\circ, 20\,^\circ\}$
Standard deviation[10]	$\{0.0, 0.02, 0.05\}$

Table 3.1: Transformation parameters used to generate initialisations for the single-frame synthetic segmentation problems.

2. `PERPENDICULAR-FREE`: Single-frame fit with E^f_{FIT} from (3.26), *free* correspondences, and perpendicular boundary candidate selection.

3. `UNCONSTRAINED`: Single-frame fit with E^f_{FIT} from (3.26), free correspondences, and *unconstrained* boundary candidate selection (§3.2.4).

4. `PAIRWISE`: `UNCONSTRAINED` with pairwise boundary candidate constraints (§3.2.6).

For each algorithm, boundary candidate selection and model contour refinement were repeated in alternation for a fixed number of iterations, with the correspondences initialised uniformly along the length of the model contour at each iteration. Continuous optimisation for all algorithms was terminated if $\mu < 10^{-9}$, up to a maximum of 200 `LM` iterations.

For each segmentation problem, each algorithm was run multiple times with different initial model contours to assess its robustness to initialisation. Segmentation accuracy was measured by the Hausdorff distance [8] between the final model contour and a ground-truth contour.

Synthetic Edge Maps Multiple initial model contours were generated for each of the five synthetic segmentation problems by scaling, translating, rotating and applying random displacements to the test contour control points. Specifically, scaling and rotation were applied about the centroid of the control points, and the random displacements were drawn from an isotropic zero-mean Gaussian. A total of 243 initialisations were generated with the transformation parameters given in Table 3.1. The original test contours were used as the ground-truth contours when measuring segmentation accuracy. Consistent and constant model and optimisation settings were used for all algorithms across all initialisations and synthetic edge maps (Table 3.2).

[10] A standard deviation of zero is used here with slight mathematical abuse to indicate no noise.

[11] "Iterations" is the number of rounds of alternation of boundary candidate selection and model contour fitting.

[12] For `PERPENDICULAR-FIXED` and `PERPENDICULAR-FREE`, only boundary candidates that are within a distance of the "normal length" of the model contour are retained (§3.1.3, (3.1)).

λ_η	$\lambda_\mathcal{R}$	λ_P	$N_{\boldsymbol{u}}$	Iterations[11]	Normal length[12]
0.2	0.05	10.0	100	10	0.375

Table 3.2: Parameter settings for `PERPENDICULAR-FIXED`, `PERPENDICULAR-FREE`, `UNCONSTRAINED` and `PAIRWISE` for the single-frame synthetic segmentation problems.

For 10 iterations, typical execution times were: `PERPENDICULAR-FIXED` 0.9s, `PERPENDICULAR-FREE` 1.1s, `UNCONSTRAINED` 1.3s, and `PAIRWISE` 1.9s.

The segmentation errors for each algorithm applied to `CONVEX-8`, `CONCAVE-8`, and `CONCAVE-12` are shown in Figure 3.12.

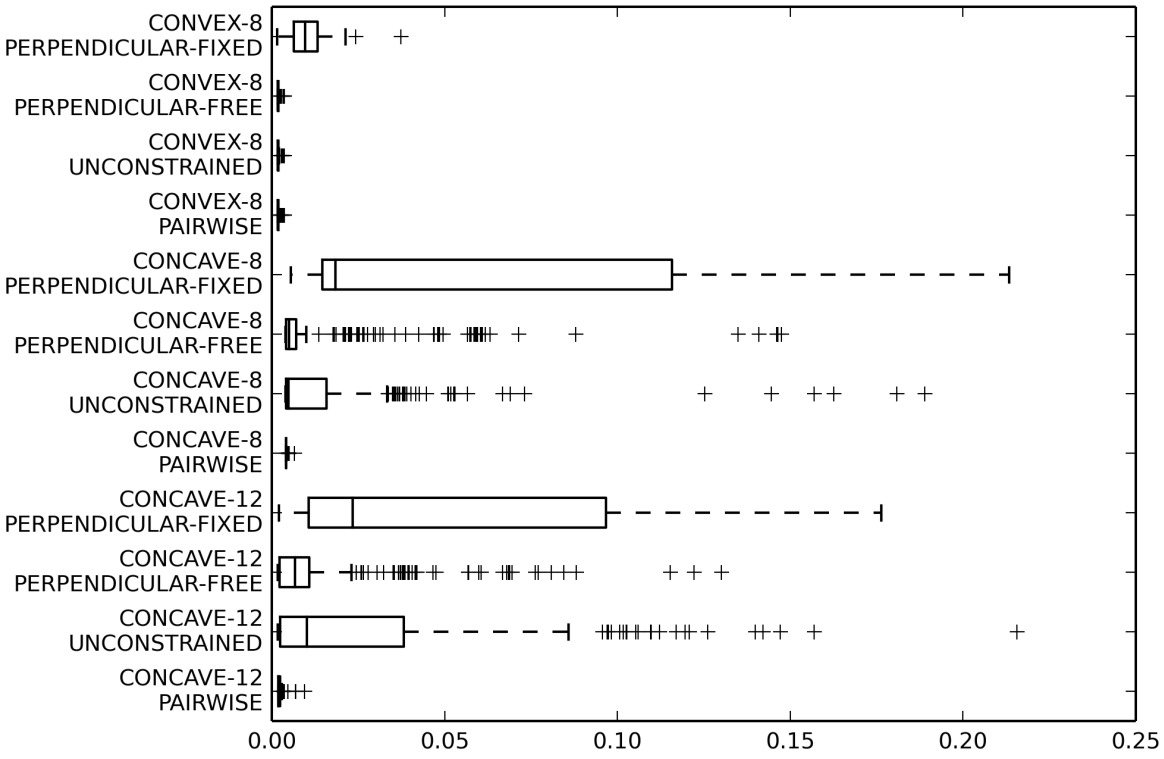
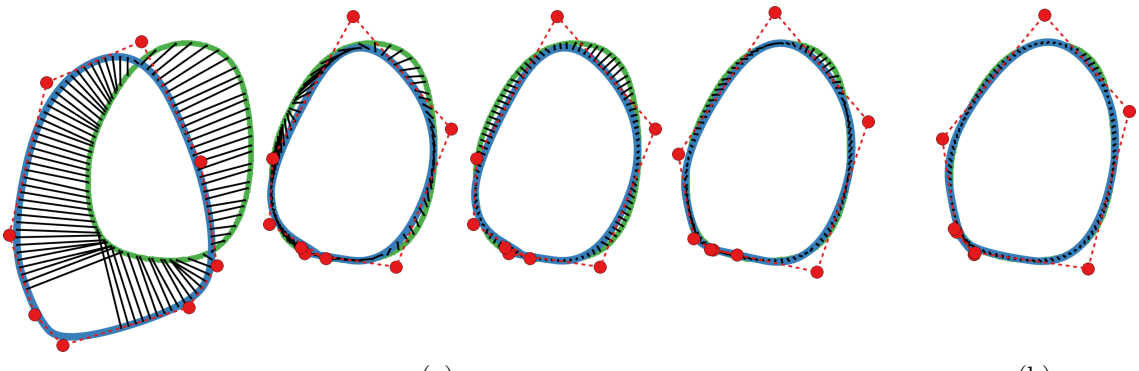


Figure 3.12: Boxplots of segmentation errors (no units) for `PERPENDICULAR-FIXED`, `PERPENDICULAR-FREE`, `UNCONSTRAINED`, and `PAIRWISE` applied to `CONVEX-8`, `CONCAVE-8`, and `CONCAVE-12`.

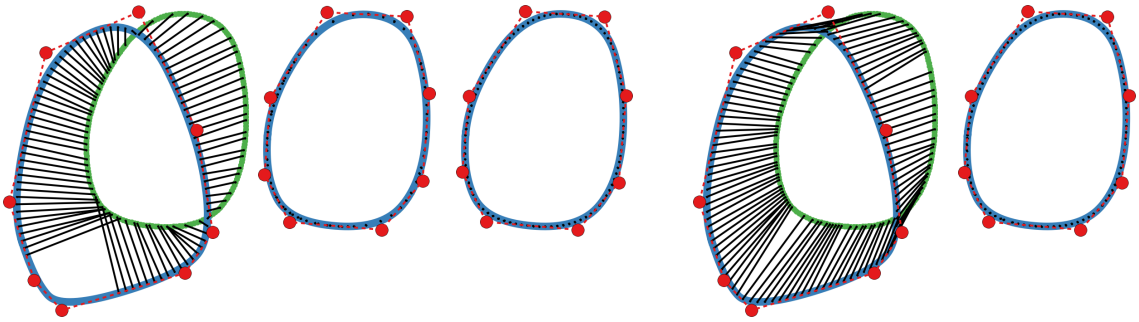
For `CONVEX-8`, `PERPENDICULAR-FIXED` has the worst segmentation performance. Intermediate steps of `PERPENDICULAR-FIXED` are shown for the initialisation with the maximum inlier segmentation error in Figure 3.13. Importantly, due to the perpendicular boundary candidate constraint, the "connections" from the model contour to the boundary candidates "cross" after the first step of discrete optimisation. Since the model correspondences are fixed, the control points at the base of the model contour become compressed to minimise the fitting error, causing the model contour to

self-intersect and restricting its flexibility in subsequent iterations. However, when the correspondences are free this does not occur (Figure 3.14a) because the correspondences move across the model contour to correct and "uncross" the connections. Importantly, incorrect boundary candidate selection is avoided altogether when the perpendicular boundary candidate constraint is removed (Figure 3.14c).



(a) (b)

Figure 3.13: Intermediate steps of `PERPENDICULAR-FIXED` on `CONVEX-8` for the initialisation with the maximum inlier segmentation error (2.37×10^{-2}). The model contour (blue), control points (red) and correspondences (black) are shown on top of the boundary candidates (green). (a) Intermediate results of two alternating steps of boundary candidate selection and model contour fitting. (b) Final model contour.

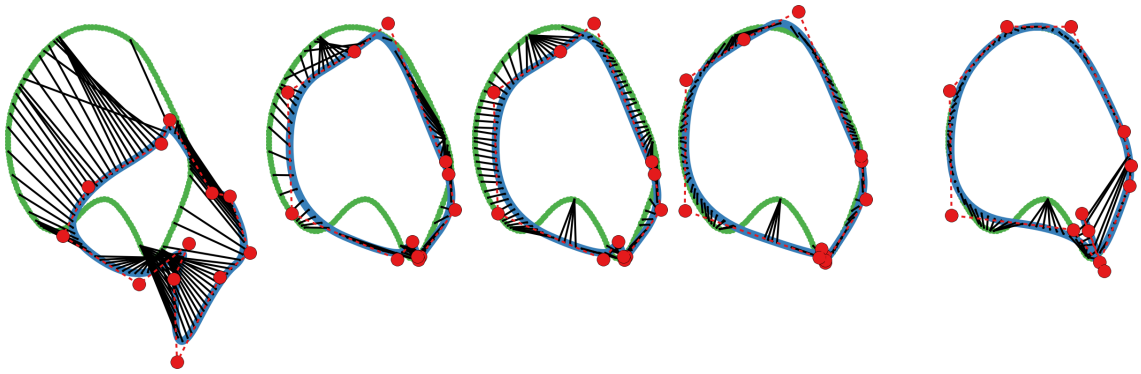


(a) (b) (c) (d)

Figure 3.14: (a) Single iteration of boundary candidate selection and model contour fitting for `PERPENDICULAR-FREE` on `CONVEX-8` with the same initialisation as Figure 3.13. (b) Final model contour (1.80×10^{-3}). (c) Boundary candidate selection for `UNCONSTRAINED`. (d) Final model contour (1.76×10^{-3}).

Figure 3.12 also shows that the segmentation performance of all algorithms is worse for `CONCAVE-8` and `CONCAVE-12` compared to `CONVEX-8`. Comparing upper quartile positions, `PERPENDICULAR-FIXED` performs the worst, followed by `UNCONSTRAINED`, `PERPENDICULAR-FREE` and `PAIRWISE`. The increase in segmentation error

for PERPENDICULAR-FIXED, UNCONSTRAINED and PERPENDICULAR-FREE is due to the concave section at the base of the contour which inhibits independent boundary candidate selection. By example, Figure 3.15 shows the intermediate steps for UNCONSTRAINED on CONCAVE-12 for the result with the maximum inlier segmentation error. As shown, boundary candidates are incorrectly selected which causes contraction of the model contour control points and prevents accurate delineation of the concave section in subsequent iterations. PERPENDICULAR-FREE performs similarly (Figures 3.16a and 3.16b). However, by including pairwise constraints and selecting boundary candidates jointly, PAIRWISE successfully fits the model contour (Figures 3.16c and 3.16d). This performance is consistent for CONCAVE-8 and CONCAVE-12 (Figure 3.12).



(a) (b)

Figure 3.15: Intermediate steps of UNCONSTRAINED on CONCAVE-12 for the initialisation with the maximum inlier segmentation error (9.19×10^{-2}). (a) Intermediate results of two alternating steps of boundary candidate selection and model contour fitting. (b) Final model contour.

The segmentation errors for each algorithm applied to CONVEX-8-MB and CONCAVE-8-MB are shown in Figure 3.17.

As before, PERPENDICULAR-FIXED performed the worst, PERPENDICULAR-FREE and UNCONSTRAINED achieved similar segmentation errors, and PAIRWISE had consistently low segmentation errors. However, one noticeable exception is the outlier with error 6.35×10^{-1} for PAIRWISE on CONCAVE-8-MB. The intermediate iterations in Figure 3.18 show the degenerate segmentation result. In effect, the combination of the pairwise constraints and poor initialisation of the model contour result in a tight loop of boundary candidates being selected preferentially over a less focused and more desirable selection. For comparison, steps for the result with the median segmentation error are shown in Figure 3.19. As shown, accurate segmentation is achieved even when missing boundaries are not specifically modelled. Although correspondences

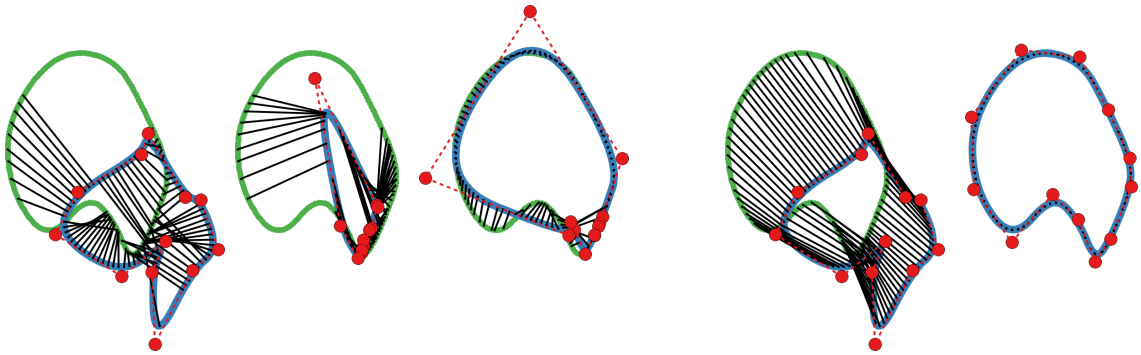


Figure 3.16: (a) Single iteration of boundary candidate selection and model contour fitting for PERPENDICULAR-FREE on CONCAVE-12 with the same initialisation as Figure 3.15. (b) Final model contour (8.09×10^{-2}). (c) Boundary candidate selection for PAIRWISE. (d) Final model contour (2.36×10^{-3}).

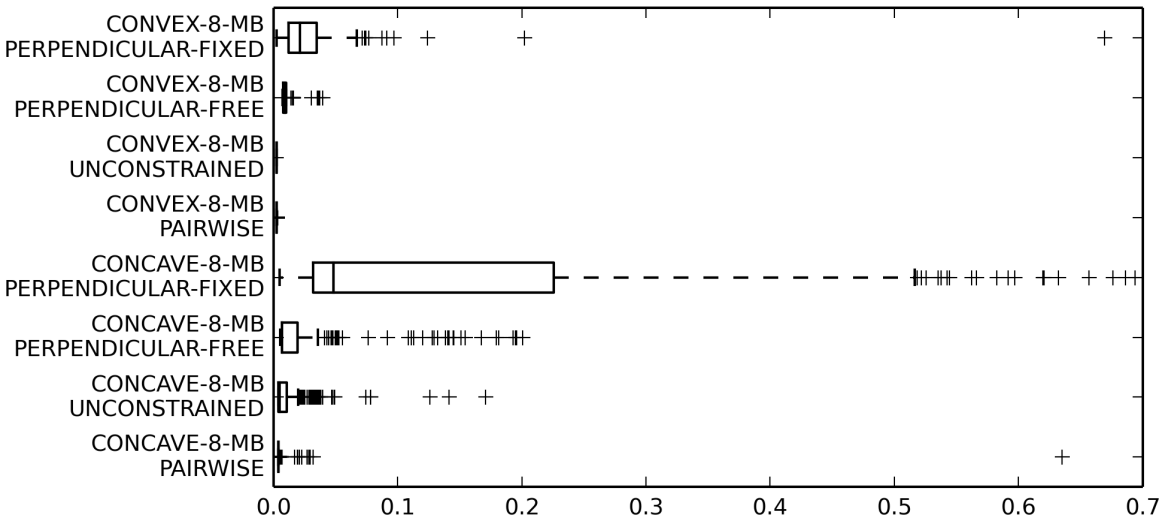


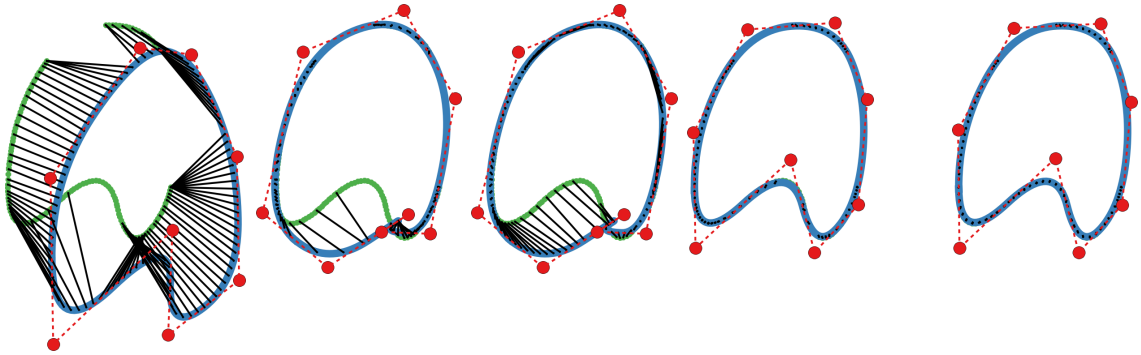
Figure 3.17: Boxplots of segmentation errors (no units) for PERPENDICULAR-FIXED, PERPENDICULAR-FREE, UNCONSTRAINED, and PAIRWISE algorithms applied to CONVEX-8-MB and CONCAVE-8-MB.

without natural boundary candidates form connections far from their initial model point, the correspondences move along the model contour to compensate when jointly optimised with the model contour control points.

2D Echocardiography Slices Similar to the synthetic edge maps, multiple initial model contours were generated for each of the 10 long-axis ultrasound slices by translating and randomly perturbing the ground-truth model contour (Table 3.3). Scale and rotation transformations were omitted to simplify the analysis so that only

(a) (b)

Figure 3.18: Intermediate steps of PAIRWISE on CONCAVE-8-MB for the initialisation with the maximum outlier segmentation error (6.35×10^{-1}). (a) Intermediate results of two alternating steps of boundary candidate selection and model contour fitting. (b) Final model contour.



(a) (b)

Figure 3.19: Intermediate steps of PAIRWISE on CONCAVE-8-MB for the initialisation with the median segmentation error (3.89×10^{-3}). (a) Intermediate results of two alternating steps of boundary candidate selection and model contour fitting. (b) Final model contour.

9 initialisations were generated for each long-axis slice.

As before, consistent parameter settings were used for all algorithms (Table 3.4). With comparison to Table 3.2, three parameters are different: λ_P, "Iterations", and "Normal length". Normal length was necessarily updated for the change in physical coordinate system from the synthetic edge maps to the echocardiography slices. Fewer iterations were also deemed sufficient for the algorithms to either achieve a good model fit, or fail outright. Finally, λ_P was reduced to reflect the increase in uncertainty of pairwise coherency as a result of the spurious edge responses that arise in the left ventricle blood pool.

The segmentation errors for all algorithms are shown in Figure 3.20.

Translation (mm)	$\{(x, y) : x \in \{-5, 0, 5\}, y \in \{-5, 0, 5\}\}$
Standard deviation (mm)	$\{2.5\}$

Table 3.3: Transformation parameters used to generate initialisations for the single-frame echocardiography segmentation problems.

λ_η	$\lambda_\mathcal{R}$	λ_P	$N_{\boldsymbol{u}}$	Iterations	Normal length (mm)
0.2	0.05	2.0	100	3	7.5

Table 3.4: Parameter settings for `PERPENDICULAR-FIXED`, `PERPENDICULAR-FREE`, `UNCONSTRAINED` and `PAIRWISE` for the single-frame echocardiography segmentation problems.

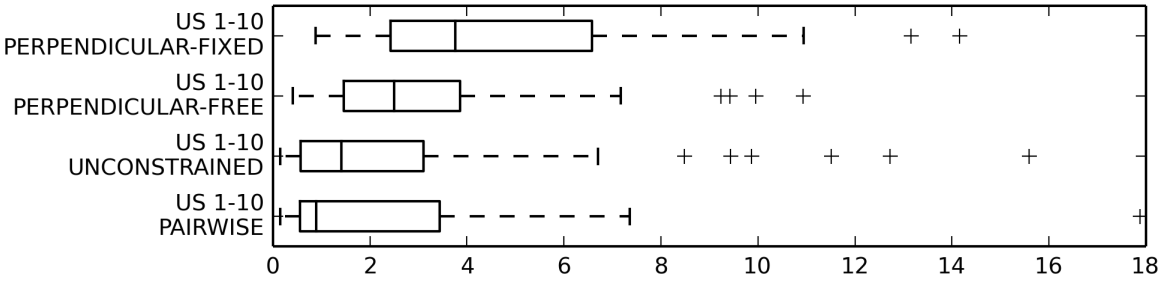


Figure 3.20: Boxplots of segmentation errors (mm) for `PERPENDICULAR-FIXED`, `PERPENDICULAR-FREE`, `UNCONSTRAINED`, and `PAIRWISE` algorithms applied to 10 long-axis ultrasound slices.

The outlier segmentation failure for `PAIRWISE` is shown in Figure 3.21. The boundary candidates corresponding to the endocardial border are "short-circuited" due to the combination of the poor initialisation and spurious interior edge responses. For comparison, the median segmentation result for `PAIRWISE` is shown in Figure 3.22.

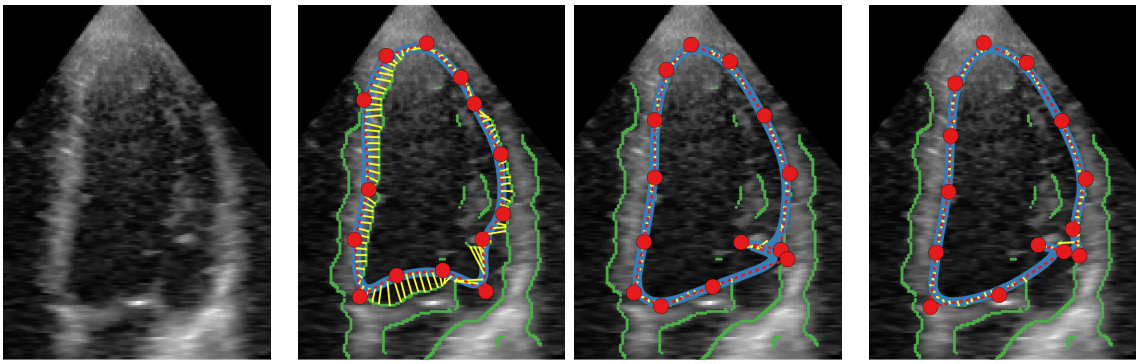
Example segmentation failures for `PERPENDICULAR-FIXED` are shown in Figures 3.23 and 3.24. Importantly, `PAIRWISE` successfully segments both frames given the same model contour initialisation.

3.4.4 2D+t Segmentation

Four segmentation algorithms were compared for 2D+t segmentation:

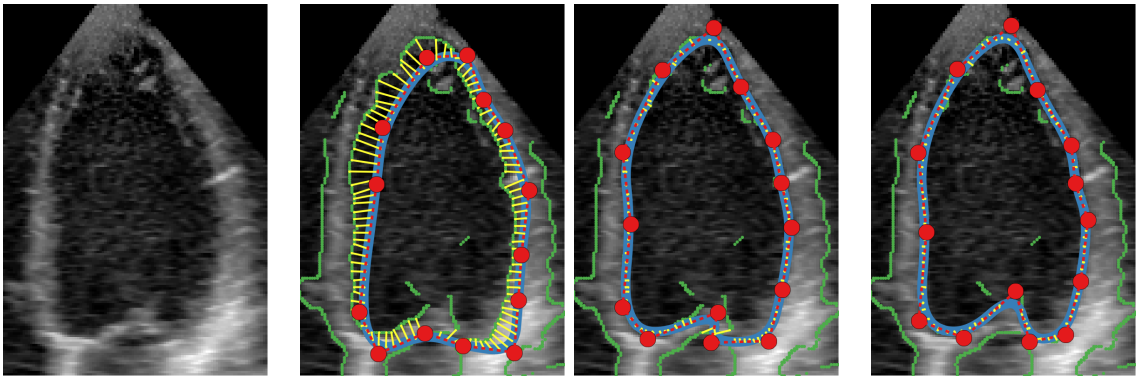
1. `SEQUENTIAL-FIXED`: The standard "track-to-last" algorithm (§3.1.6) with frame-by-frame fitting, fixed correspondences and perpendicular boundary candidate selection (3.1).

2. `SHARED`: Alternating unconstrained *joint* optimisation of $E_{\textsc{Shared-P}}$ (3.14) with *free* and *shared* correspondences (§3.2.5.1).



(a) (b) (c)

Figure 3.21: Intermediate steps of `PAIRWISE` for the initialisation and long-axis echocardiography slice with the maximum outlier segmentation error (1.79×10^{1}mm). The model contour (blue), control points (red) and correspondences (yellow) are shown on top of the boundary candidates (green) and superimposed on the echocardiography slice. (a) Input echocardiography slice. (b) Intermediate results of one step of boundary candidate selection and model contour fitting. (c) Final model contour.

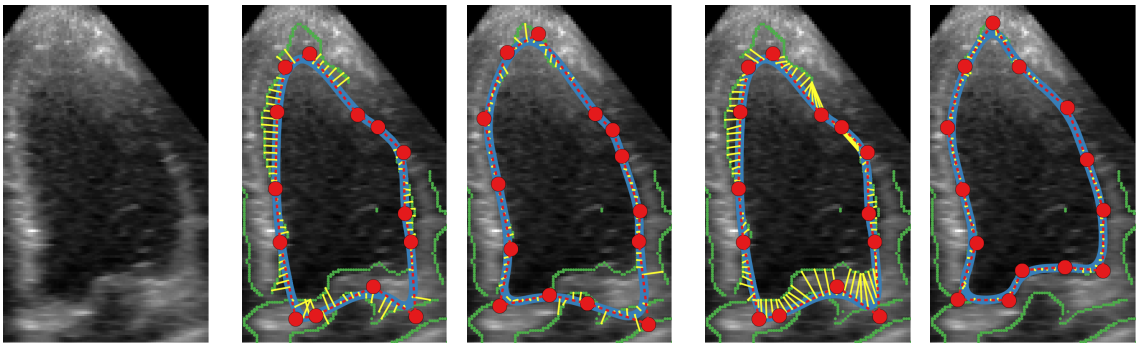


(a) (b) (c)

Figure 3.22: Intermediate steps of `PAIRWISE` for the initialisation and long-axis echocardiography slice with the median segmentation error (8.88×10^{-1}mm). (a) Input echocardiography slice. (b) Intermediate results of one step of boundary candidate selection and model contour fitting. (c) Final model contour.

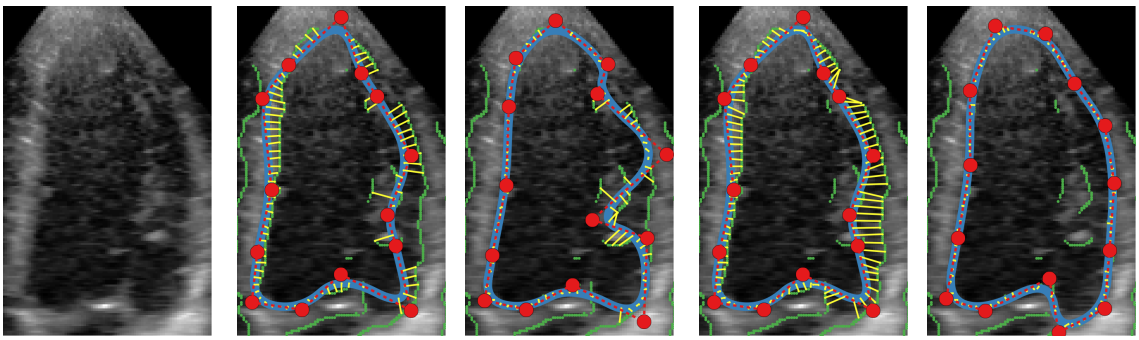
3. `SEPARATE`: Alternating unconstrained joint optimisation of $E_{\text{SEPARATE-P}}$ (3.15) with free and *separate* correspondences (§3.2.5.2).

4. `SEQUENTIAL-FREE`: Identical to `SEQUENTIAL-FIXED` except with *free* correspondences which are optimised jointly with the model contour control points. Importantly, this algorithm does *not* guarantee anatomical consistency (§3.2.5) but it is included here for demonstrative purposes.



(a) (b) (c) (d) (e)

Figure 3.23: (a) Input echocardiography slice. (b) Initial boundary candidate selection for `PERPENDICULAR-FIXED`. (c) Final model contour for `PERPENDICULAR-FIXED` (1.32×10^{1}). (d) Initial boundary candidate selection for `PAIRWISE`. (e) Final model contour for `PAIRWISE` (6.12×10^{-1}).



(a) (b) (c) (d) (e)

Figure 3.24: (a) Input echocardiography slice. (b) Initial boundary candidate selection for `PERPENDICULAR-FIXED`. (c) Final model contour for `PERPENDICULAR-FIXED` (1.05×10^{1}). (d) Initial boundary candidate selection for `PAIRWISE`. (e) Final model contour for `PAIRWISE` (3.76×10^{-1}).

As before, boundary candidate selection and model contour refinement were repeated in alternation for a fixed number of iterations and correspondences were initialised uniformly along the length of the model contour at each iteration. Similarly, continuous optimisation for all algorithms was terminated if $\mu < 10^{-9}$, up to a maximum of 200 `LM` iterations. For `SHARED` and `SEPARATE`, discrete minimisation of each pair of frames was repeated 20 times.

For each synthetic edge map segmentation problem, each algorithm was run multiple times with different initial model contours. Anatomical consistency was measured by the maximum difference in frame-to-frame displacements between the recovered model contours and the ground-truth. Specifically, let Y^f (X^f) and Y^{f+1}

Translation	$\{(-0.05, 0), (0, -0.05), (0, 0), (0, 0.05), (0.05, 0)\}$

Table 3.5: Transformation parameters used to generate the initialisations for the 2D+t synthetic segmentation problems.

(X^{f+1}) denote the matrices of control points for the ground-truth (recovered) model contours in frames f and $f+1$. Now, let $\boldsymbol{t}$ ($\boldsymbol{u}$) denote a vector of $N_{\boldsymbol{t}}$ ($N_{\boldsymbol{u}}$) model coordinates which are uniformly sampled along the length of Y^f (X^f). Let $\boldsymbol{j}$ denote a vector of $N_{\boldsymbol{u}}$ indices where the i^{th} entry in $\boldsymbol{j}$ gives the index of the closest sample of Y^f to sample i from X^f:

$$j_i = \arg\min_k \left\| \chi\left(t_k, Y^f\right) - \chi\left(u_i, X^f\right) \right\|^2$$

Finally, let $\boldsymbol{d}$ denote the vector of N_u differences in frame-to-frame displacements between the two pairs of contours. Each d_i is given by:

$$d_i = \left\| \left(\chi\left(u_i, X^{f+1}\right) - \chi\left(u_i, X^f\right)\right) - \left(\chi\left(t_{j_i}, Y^{f+1}\right) - \chi\left(t_{j_i}, Y^f\right)\right) \right\|$$

For a given frame, the maximum difference in frame-to-frame displacements is then given by the maximum element in $\boldsymbol{d}$.

Ground-truth *anatomically consistent* model contours were not available for the 2D+t echocardiography slices which prevented quantitative assessment.

Synthetic Edge Maps Multiple initial model contours were generated for each of the two synthetic segmentation problems by translating the test contour control points from a single source frame (Table 3.5). The first frame was used to initialise `SEQUENTIAL-FIXED` and `SEQUENTIAL-FREE`; the centre frame was used for `SHARED` and `SEPARATE`. Scale, rotation and random perturbations were not included as the purpose of the experiment was to assess the anatomical consistency of the recovered model contours, *not* to test robustness to initialisation. Consistent and constant model and optimisation settings were used for all algorithms across all initialisations and synthetic edge maps. The single-frame settings for λ_η, $\lambda_\mathcal{R}$, λ_P and normal length were used as given in Table 3.2; the additional parameter settings are given in Table 3.6.

For `SHARED` applied to `CONCAVE-10`, continuous optimisation of 2660 parameters ($2 \times 10 \times 23 + 1 \times 100 \times (23 - 1) = 2660$) with 18060 residuals took approximately 5s per 100 iterations. `SEPARATE` took approximately 9s per 100 iterations for 4860 parameters ($2 \times 10 \times 23 + 1 \times 100 \times 2 \times (23 - 1) = 4860$) with 26860 residuals. For both algorithms, discrete optimisation of each pair of frames took approximately 2.5s with 20 restarts. By comparison, a single round of boundary candidate selection *and* model

λ_G	$\lambda_{\mathcal{A}}$	λ_C	Iterations	Search radius[13]
2.0	0.1	0.5	5	0.05

Table 3.6: Parameter settings (in addition to Table 3.2) for `SEQUENTIAL-FIXED`, `SHARED`, `SEPARATE` and `SEQUENTIAL-FREE` for the 2D+t synthetic segmentation problems.

contour fitting for `SEQUENTIAL-FIXED` (`SEQUENTIAL-FREE`) took only 80ms (110ms) per frame. Typical executions times for 5 complete iterations across 22 frames were: `SEQUENTIAL-FIXED` 10s, `SHARED` 280s, `SEPARATE` 300s, `SEQUENTIAL-FREE` 12s.

The segmentation errors for each algorithm applied to `CONVEX-8` and `CONCAVE-10` are shown in Figure 3.25.

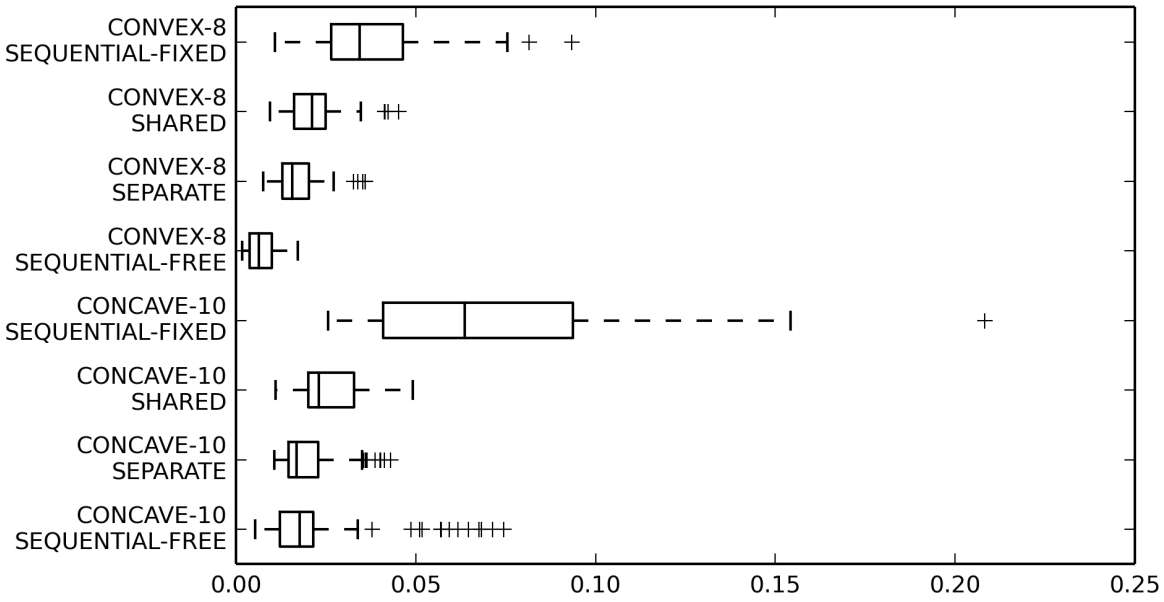
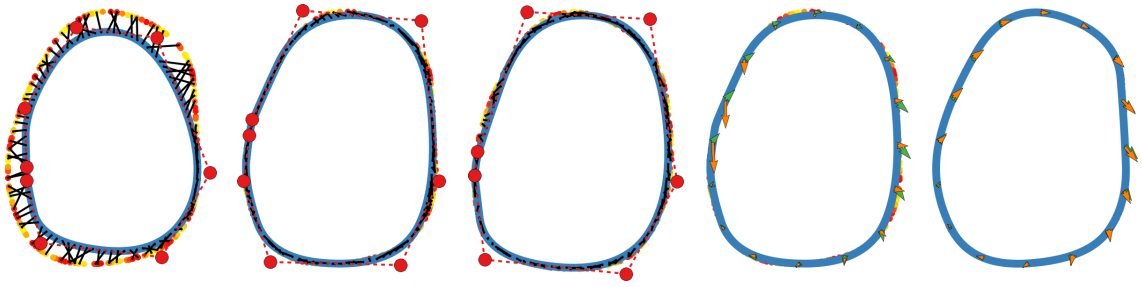


Figure 3.25: Boxplots of segmentation errors (no units) for `SEQUENTIAL-FIXED`, `SHARED`, `SEPARATE`, and `SEQUENTIAL-FREE` algorithms applied to `CONVEX-8` and `CONCAVE-10`.

For both `CONVEX-8` and `CONCAVE-10`, `SEQUENTIAL-FIXED` had the worst segmentation performance. Intermediate steps of `SEQUENTIAL-FIXED` for the frames from `CONVEX-8` with the outlier and maximum segmentation errors are shown in Figures 3.26 and 3.27. Similarly, intermediate steps for the frame with maximum segmentation error from `CONCAVE-10` is shown in Figure 3.28.

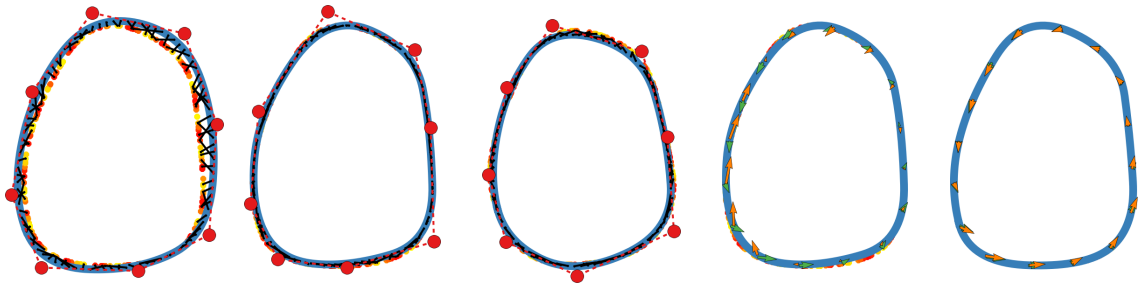
Although the recovered model contours are spatially close to the boundary candidates — especially for `CONVEX-8` — the deformations predicted by the recovered model contours do *not* match those of the ground-truth (Figure 3.26d, Figure 3.27d,

[13]For `SEQUENTIAL-FIXED` and `SEQUENTIAL-FREE` this is the radius used in the definition of Γ^f which restricts frame-to-frame boundary candidate refinement (§3.1.4, (3.4)).



(a) (b) (c) (d) (e)

Figure 3.26: Intermediate steps for SEQUENTIAL-FIXED on frame $f = 15$ of CONVEX-8 for the initialisation with the maximum outlier segmentation error (9.34×10^{-2}). (a)–(c) The model contour (blue), control points (red) and correspondences (black) are shown on top of the coloured boundary candidates (yellow-red). (d)–(e) The ground-truth (green) and recovered (orange) frame-to-frame deformations superimposed on the model contour. (a) Initial model contour for $f = 15$. (b) Final model contour for $f = 15$. (c) Final model contour for $f = 16$. (d) Final frame-to-frame deformations at $f = 15$. (e) Frame-to-frame deformations from SEPARATE (8.51×10^{-3}).



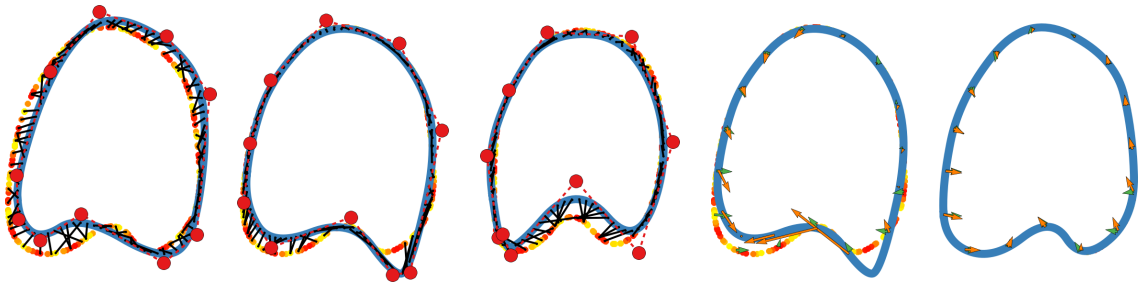
(a) (b) (c) (d) (e)

Figure 3.27: Intermediate steps for SEQUENTIAL-FIXED on frame $f = 8$ of CONVEX-8 for the initialisation with the maximum inlier segmentation error (7.56×10^{-2}). (a) Initial model contour for $f = 8$. (b) Final model contour for $f = 8$. (c) Final model contour for $f = 9$. (d) Final frame-to-frame deformations at $f = 8$. (e) Frame-to-frame deformations from SEPARATE (1.39×10^{-2}).

and Figure 3.28d). Importantly, SEPARATE predicts deformations which align closely with the ground-truth (Figure 3.26e, Figure 3.27e, and Figure 3.28e).

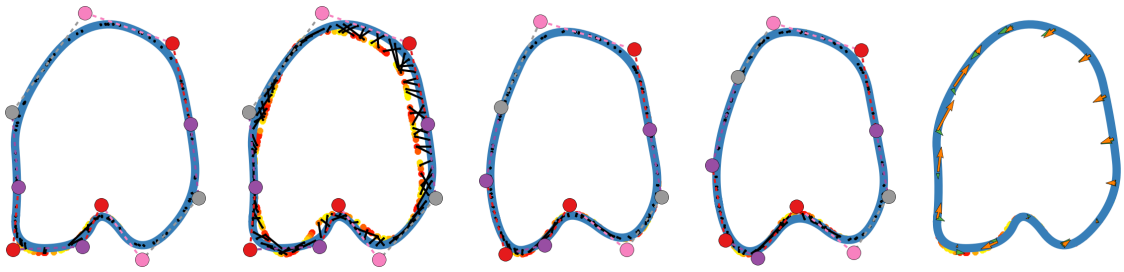
Interestingly, Figure 3.12 shows that SEQUENTIAL-FREE has the best segmentation performance for CONVEX-8 and similar performance to SHARED and SEPARATE for CONCAVE-10. This result appears to contradict the argument that SEQUENTIAL-FREE does *not* guarantee anatomical consistency, but both the favourable and unfavourable results are easily explained.

First, model contours recovered by SEQUENTIAL-FREE *may* still be anatomically



(a) (b) (c) (d) (e)

Figure 3.28: Intermediate steps for `SEQUENTIAL-FIXED` on frame $f = 7$ of `CONCAVE-10` for the initialisation with the maximum inlier segmentation error (1.54×10^{-1}). (a) Initial model contour for $f = 7$. (b) Final model contour for $f = 7$. (c) Final model contour for $f = 8$. (d) Final frame-to-frame deformations at $f = 7$. (e) Frame-to-frame deformations from `SEPARATE` (3.59×10^{-2}).



(a) (b) (c) (d) (e)

Figure 3.29: Intermediate steps for `SEQUENTIAL-FREE` on frame $f = 9$ of `CONCAVE-10` for the initialisation with the maximum outlier segmentation error (7.44×10^{-2}). Control points are coloured red, purple, gray and pink. (a) Final model contour for $f = 9$. (b) Initial boundary candidate selection for $f = 10$. (c) Initial fit of model contour for $f = 10$. (d) Final model contour for $f = 10$. (e) Final frame-to-frame deformations at $f = 9$.

consistent if the ground-truth deformation is small *and* the frame-to-frame search radius is small (§3.1.4). However, this is not guaranteed as shown by the failure from `CONCAVE-10` in Figure 3.29. Similarly, Figure 3.30 shows that the segmentation errors for both `SEQUENTIAL-FIXED` and `SEQUENTIAL-FREE` increase dramatically when the frame-to-frame search radius is doubled to 0.10. As evidenced by Figure 3.31, the size of the search radius provides a hard constraint on which boundary candidates can be selected and it must be set precisely even when the ground-truth deformations are small. By contrast, decreasing λ_G by a factor of four — reducing the weight that boundary candidates in adjacent frames must be spatially close — results in nearly identical segmentation performance for both `SHARED` and `SEPARATE` (Figure 3.32).

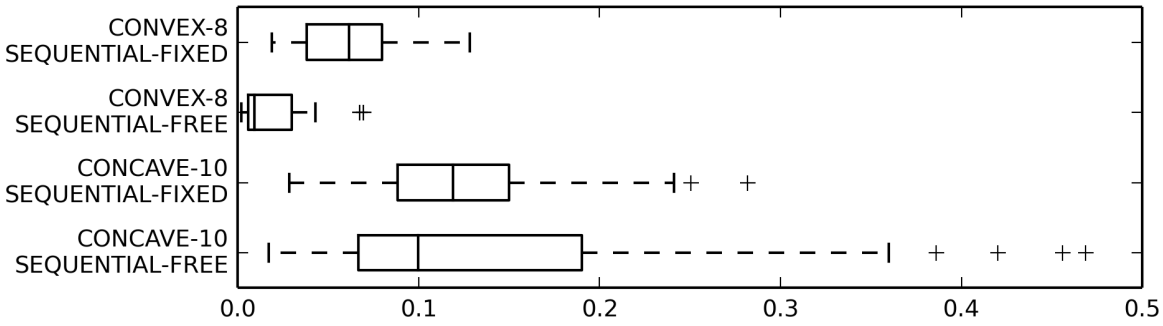


Figure 3.30: Boxplots of segmentation errors (no units) for `SEQUENTIAL-FIXED` and `SEQUENTIAL-FREE` with the frame-to-frame search radius doubled to 0.10. (Note the x-axis scale difference to Figure 3.25.)

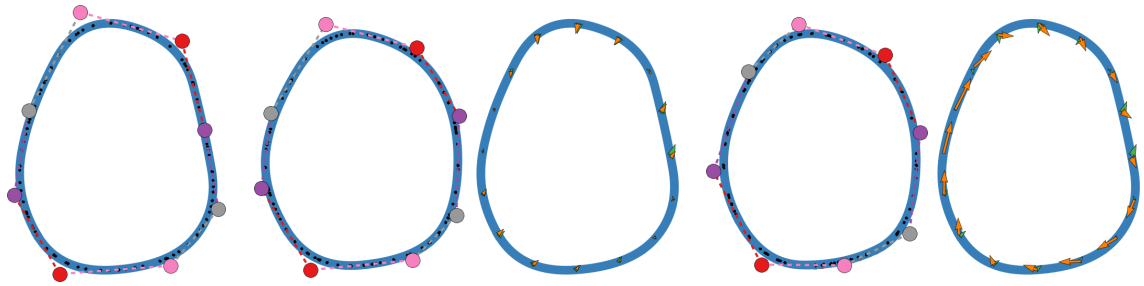


Figure 3.31: A demonstration of `SEQUENTIAL-FREE` failure when the appearance search radius is too large (`CONVEX-8`). (a) Final model contour for $f = 9$ (frame-to-frame search radius of 0.05). (b) Final model contours for $f = 10$ (search radius of 0.05). (c) Final model contours for $f = 10$ (search radius of 0.10). Tangential deformation is introduced by the movement of the left gray control point towards the top of the frame.

Second, the reason that `SEQUENTIAL-FREE` can achieve superior segmentation results — especially for `CONVEX-8` — is precisely because model coordinate differences of selected boundary candidates in adjacent frames are completely ignored. That is, if correct boundary candidates in adjacent frames are selected which are anatomically consistent *and* spatially close, a superior segmentation *may* result because completely unconstrained correspondences enable a tighter model fit. However, there is no guarantee of this performance and it is very dependent on the level of regularisation. Figure 3.33 shows the segmentation errors for `SHARED`, `SEPARATE`, and `SEQUENTIAL-FREE` with $\lambda_{\mathcal{R}}$ decreased by a factor of 50. As shown, segmentation performance improves for `SEQUENTIAL-FREE` on `CONVEX-8` but worsens dramatically for `CONCAVE-10`. As before, performance of `SHARED` and `SEPARATE` remains consistent.

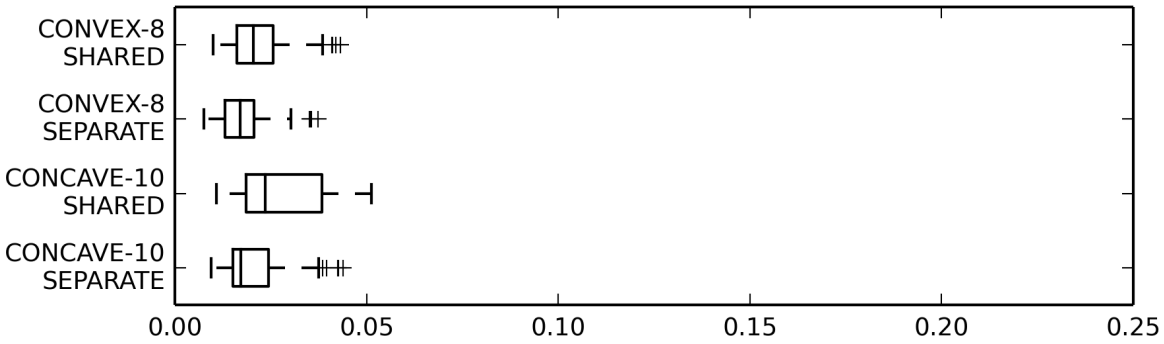


Figure 3.32: Boxplots of segmentation errors (no units) for `SHARED` and `SEPARATE` with λ_{G} decreased by a factor of four from 2.0 to 0.5. (Note the x-axis scale difference to Figure 3.30.)

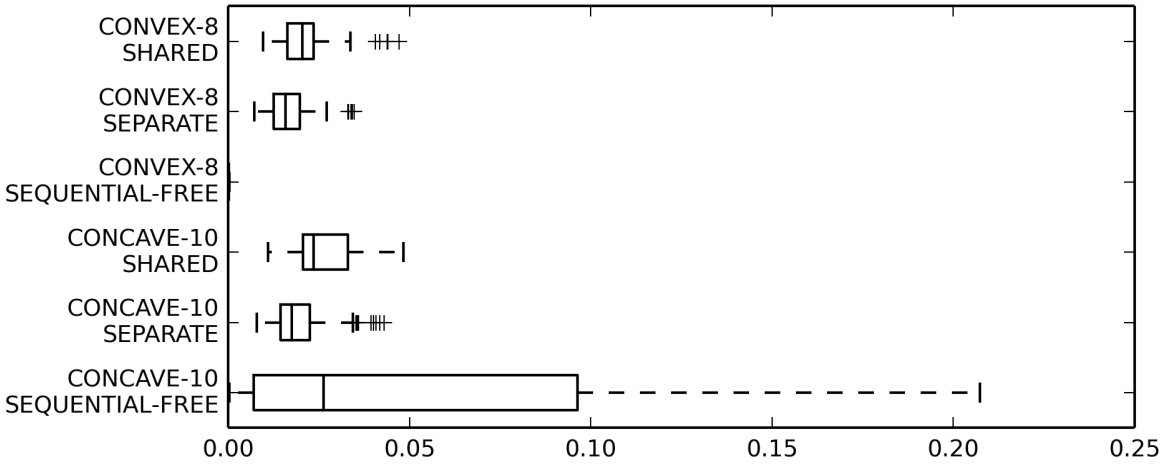


Figure 3.33: Boxplots of segmentation errors (no units) for `SHARED`, `SEPARATE` and `SEQUENTIAL-FREE` with $\lambda_{\mathcal{R}}$ decreased by a factor of 50 from 0.05 to 0.001. (Outliers omitted for `SEQUENTIAL-FREE` at 0.37, 0.40, 0.46.)

λ_{G}	$\lambda_{\mathcal{A}}$	Iterations
0.5	128.0	1

Table 3.7: Parameter settings (in addition to Tables 3.2 and 3.6) for SEPARATE for the 2D+t echocardiography segmentation examples.

2D Echocardiography Slices Ground-truth deformations were not available for the 2D+t echocardiography slices because manually setting anatomically consistent model contours is unreliable and error-prone. Quantitative segmentation assessment for the 2D+t echocardiography sequences was therefore not possible. However, a demonstration of SEPARATE is provided for qualitative assessment.

Identical model and optimisation parameters to those used for the 2D+t synthetic experiments were used except for the changes given in Table 3.7. $\lambda_{\mathcal{A}}$ was necessarily updated for the change in appearance from the synthetic edge maps to the echocardiography slices. A single iteration of SEPARATE was deemed visually sufficient because the initialisations were closer to the endocardial boundary than in previous experiments. λ_{G} was reduced by a factor of four to 0.5 based on the performance of SEPARATE on the synthetic sequences.

Initialisations, segmentation steps, and results are shown for a subset of frames for two example 2D+t echocardiography sequences in Figures 3.34 and 3.35. Model contours were deliberately initialised poorly for each frame of each sequence, as shown in Figures 3.34a, 3.34b, 3.35a, and 3.35b. After one iteration of SEPARATE, the control points are updated so that the final model contours delineate the endocardium border (Figures 3.34c, 3.34d, 3.35c, and 3.35d). By visual inspection of adjacent frames, the control points appear to move to similar anatomical positions. For example, in Figure 3.34 the red control point near the top of the frame moves to the apex and the purple control point at the bottom tracks the mitral valve. As a result, the recovered deformations of the myocardium, mitral valve, and apex appear plausible (Figures 3.36a, 3.36b, 3.37a, and 3.37b).

Deformations recovered from SEPARATE *without* frame-to-frame geometric and appearance terms are shown in Figures 3.36c, 3.36d, 3.37c, and 3.37d. With $\lambda_{\mathrm{G}} = \lambda_{\mathcal{A}} = 0$, each frame is effectively processed independently and the plausibility and consistency of the recovered deformations is totally dependent on the model contour initialisation and regularisation. Comparison to SEPARATE with non-zero λ_{G} and $\lambda_{\mathcal{A}}$ enables qualitative assessment of the impact of the frame-to-frame terms in the absence of ground-truth.

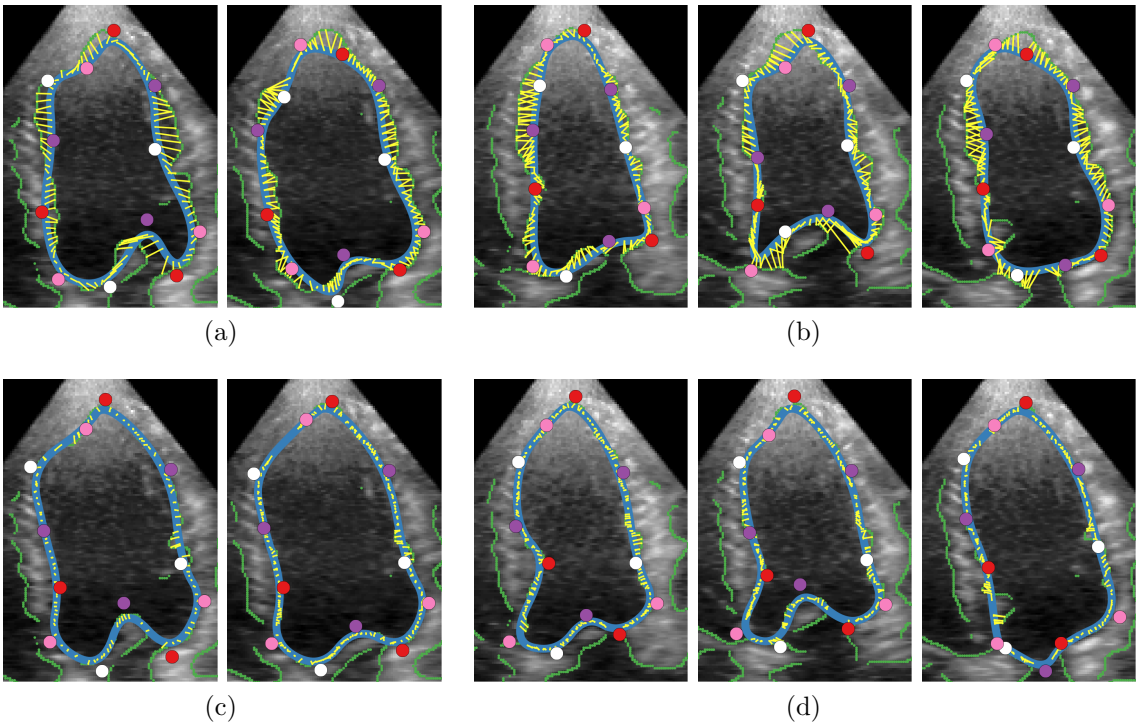


Figure 3.34: Intermediate steps of SEPARATE for a long-axis echocardiography sequence. The model contour (blue) and correspondences (yellow) are shown on top of the boundary candidates (green) and superimposed on the echocardiography slice. Control points are coloured red, purple, white, and pink (the control mesh is not shown). (a) Initial model contours and selected boundary candidates for frames 1 and 2. (b) Frames 7, 8, and 9. (c), (d) Final model contours.

With $\lambda_{\mathrm{G}} = \lambda_{\mathcal{A}} = 0$, the magnitude of the frame-to-frame deformations is overestimated due to spurious deformations tangential to the model contour present in the initialisation *not* being corrected. For example, Figure 3.36d suggests deformations along the interventricular septum which are contradictory from one frame to the next. Comparison of Figures 3.37c to 3.37a also shows deformation along the interventricular septum which is not supported when the frame-to-frame geometric and appearance terms are included. Figures 3.36c and 3.36a, and Figures 3.36d and 3.36b, also demonstrate that the frame-to-frame terms remove spurious deformation components tangential to the mitral valve.

3.5 Conclusions

In this chapter, extensions to 2D/2D+t model-based segmentation with an explicit model contour representation were described. These extensions were motivated by

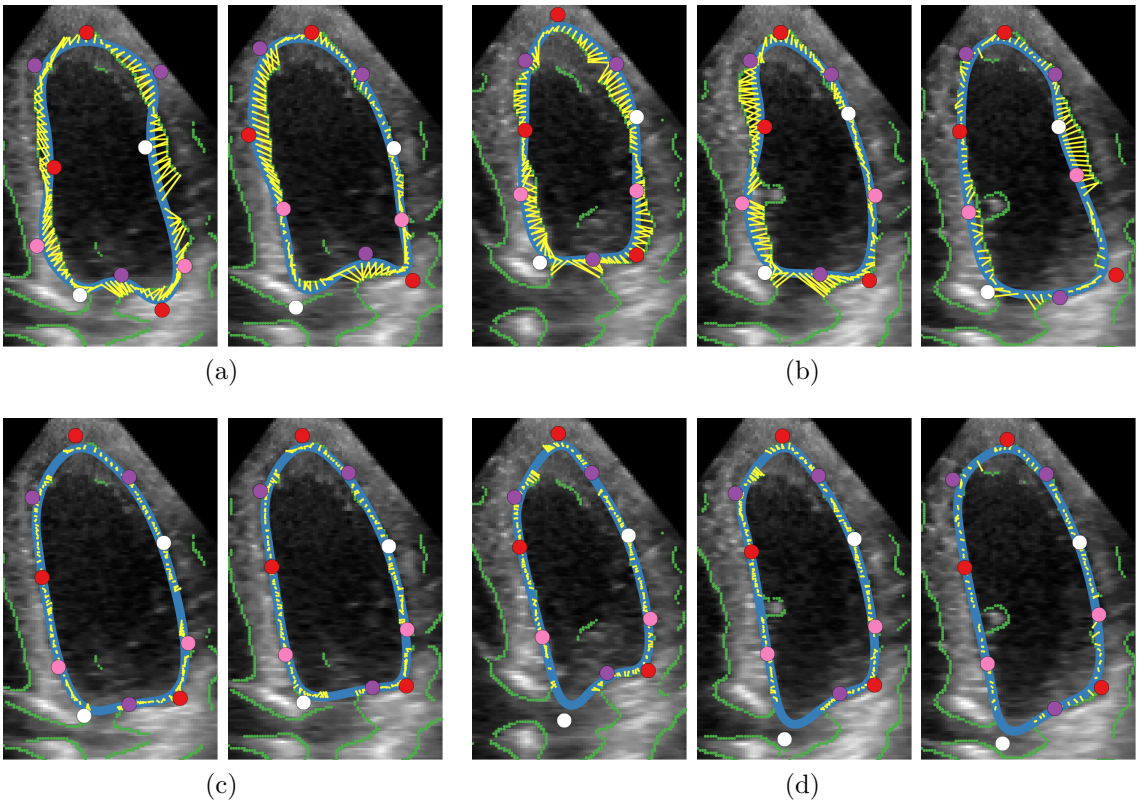


Figure 3.35: Intermediate steps of SEPARATE for a long-axis echocardiography sequence. (a) Initial model contours and selected boundary candidates for frames 3 and 4. (b) Frames 9, 10, and 11. (c), (d) Final model contours.

identification of implicit assumptions of a representative and standard 2D+t tracking formulation. Improvements to segmentation accuracy and robustness to initialisation were demonstrated by qualitative and quantitative assessment of 2D/2D+t segmentation of simulated data and long-axis echocardiography sequences.

The salient conceptual contributions of this chapter are:

1. Perpendicular boundary candidate selection is an algorithmic *choice*, not a necessity, when the model contour correspondences for each boundary candidate are optimised *jointly* with the model contour.

2. Coherent boundary candidate selection can be encouraged by introducing pairwise constraints. Notably, the discrete optimisation problem can be solved *exactly* for closed model contours in 2D using the Circular Shortest Path (CSP) algorithm.

3. For 2D+t segmentation, explicitly modelling anatomical consistency enables

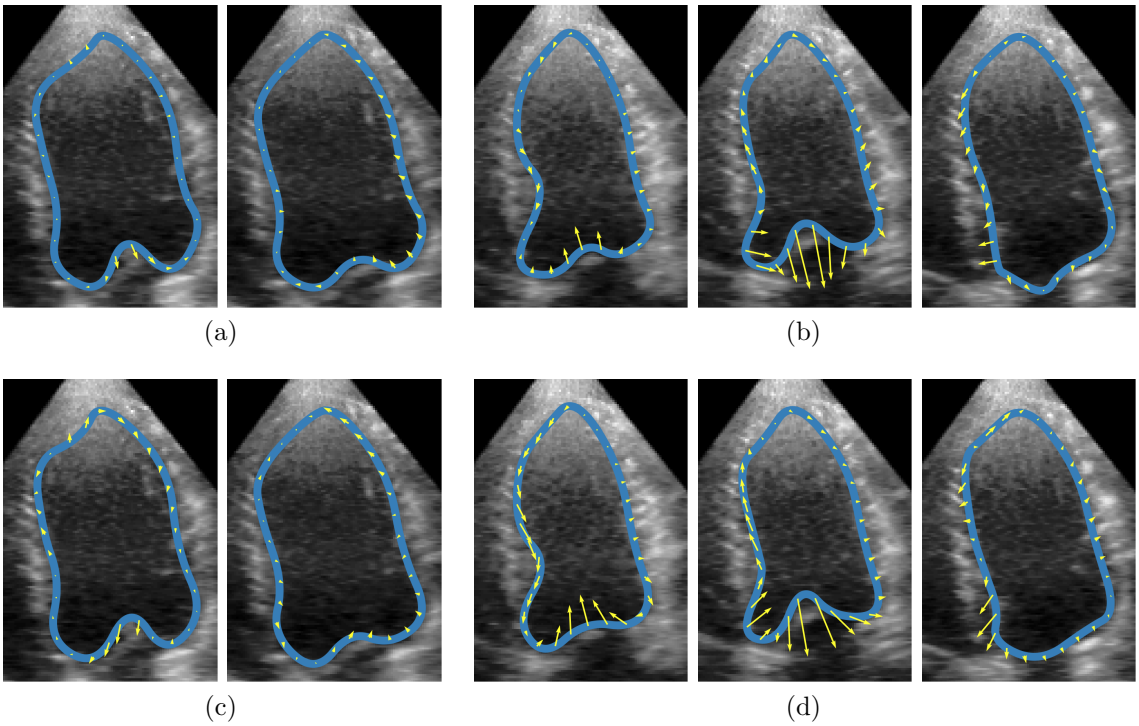


Figure 3.36: (a), (b) Recovered frame-to-frame deformations (yellow) from `SEPARATE`. (c), (d) Deformations from `SEPARATE` with $\lambda_{\mathrm{G}} = \lambda_{\mathcal{A}} = 0$.

sequential optimisation to be abandoned in favour of joint optimisation. This facilitates user-interaction and segmentation of sequences of arbitrary length.

Importantly, the concepts underpinning the extensions described in this chapter, and the associated optimisation algorithms, are general and not restricted to 2D. As will be shown in subsequent chapters, extension to 3D only requires exchanging the model contour for a model surface and generalising model coordinates to 2D manifolds.

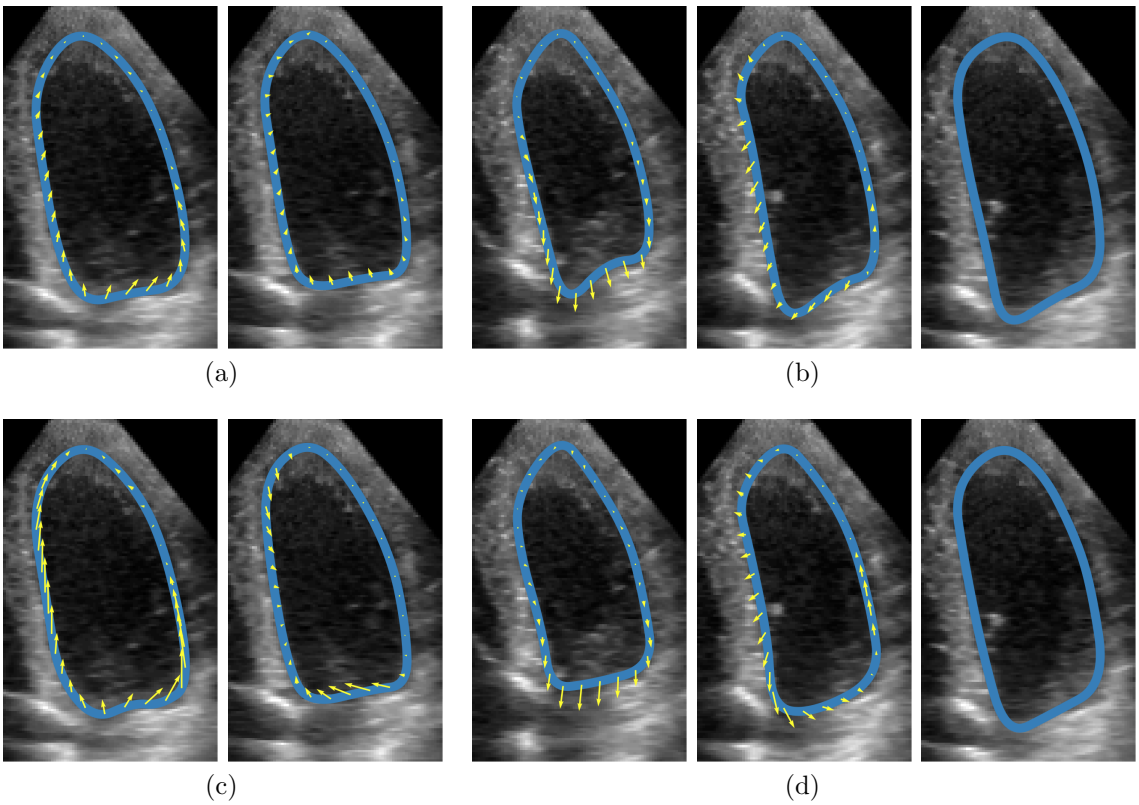


Figure 3.37: (a), (b) Recovered frame-to-frame deformations (yellow) from `SEPARATE`. (c), (d) Deformations from `SEPARATE` with $\lambda_{\mathrm{G}} = \lambda_{\mathcal{A}} = 0$.

Chapter 4

3D Model-Based Segmentation

In this chapter, the 2D model-based segmentation approach of Chapter 3 is extended to 3D.

First, definitions for the model surface and model surface domain, which extend the model contour and model contour domain, are established. 3D extensions of uniform quadratic B-splines — quadratic triangle B-splines and biquadratic B-splines — are introduced, and then *their* generalisations — Loop and Doo-Sabin subdivision surfaces — are explained in detail.

Second, an original algorithm for performing model coordinate updates is presented which enables joint optimisation of boundary candidate correspondences and surface geometry. Boundary candidate selection and surface regularisation models are also naturally extended to 3D.

Finally, three applications of 3D model-based segmentation which support ultrasound image analysis are described and presented: skull segmentation for fetal brain image analysis; face segmentation for shape analysis, and left ventricle segmentation in echocardiography for volume measurement.

The software for evaluating and verifying the model surface functions and derivatives, and for performing model coordinate updates, was implemented entirely by the author. For the Loop subdivision surfaces, this included a Python 2.7 program which generates: (a) bespoke and optimised C code for the χ, ν, and regularisation functions (and their derivatives), (b) a consistent C++ interface, and (c) a corresponding Python extension module. While `Sympy`[1] was used to aid symbolic evaluation, common subexpression elimination was implemented separately by the author to optimise the generated C code.

[1] `http://www.sympy.org`

For the Doo-Sabin subdivision surfaces, a C++ implementation with a corresponding Python extension module was also developed which performs efficient evaluation by recursive subdivision. Further details are provided in §4.3.3.

4.1 Model Surface

The model surface is defined analogously to the model contour (§3.1.2). Let $X \in \mathbb{R}^{3\times N_X}$ denote the matrix of N_X control *vertices* which define the geometry of the open or closed parametric model surface. Additionally, let $\mathcal{T}$ denote the set of edges which define a *mesh* for the parametric surface. Positions on the continuous model surface are defined by the function $\chi(\boldsymbol{u}, X)$, where $\boldsymbol{u} = (u, v) \in \Omega$ is a model coordinate and $\Omega \subset \mathbb{R}^2$ is the model surface domain. Normal vectors — or approximate normal vectors — on the model surface are defined by the function $\nu(\boldsymbol{u}, X)$. χ and ν are implicitly defined by $\mathcal{T}$ and Ω.

The simplest definition for a model surface is a polygon mesh. Consider a regular tetrahedron with Ω and χ shown in Figure 4.1. A tetrahedron is piecewise-continuous and each surface face, or *patch*, is defined by a different continuous position function χ^i over a patch domain $\Omega_i \subset \Omega$.

Consider evaluating χ for a given $\boldsymbol{u}$. Let $(i, \boldsymbol{t}) = \Psi(\boldsymbol{u})$, where i is a (one-based) patch index, $\boldsymbol{t} = (s, t) \in \triangle = \{(s, t) : s, t \in [0, 1], s + t \leq 1\}$ is the *local* patch coordinate in the unit triangle, and Ψ maps each model coordinate to i and $\boldsymbol{t}$ (Ψ^{-1} is its inverse). The position on patch i is given by:

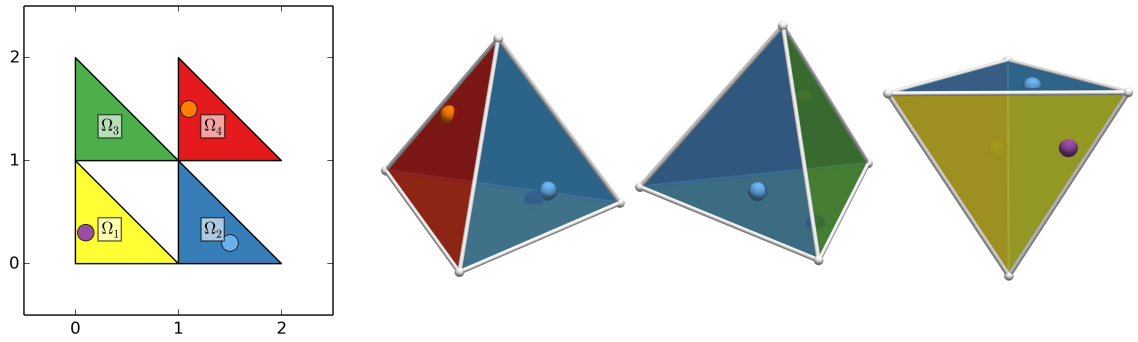
$$\chi^i(\boldsymbol{t}, X) = s\boldsymbol{x}_{\omega_1^i} + t\boldsymbol{x}_{\omega_2^i} + (1 - s - t)\,\boldsymbol{x}_{\omega_3^i}$$

where $\boldsymbol{\omega}^i$ is the vector of vertex indices for face i and is defined by $\mathcal{T}$.

Although polygon meshes, and triangle meshes in particular, are continuous and straightforward to specify, they do not have smoothly-changing normals. Fake normals can be generated (e.g. [60, 123]) but the surface remains piecewise-linear (C^0) which is undesirable for modelling naturally curved anatomical boundaries. Here, C^1 continuous generalisations of uniform quadratic B-splines (§3.3.1) in 3D are pursued as alternatives.

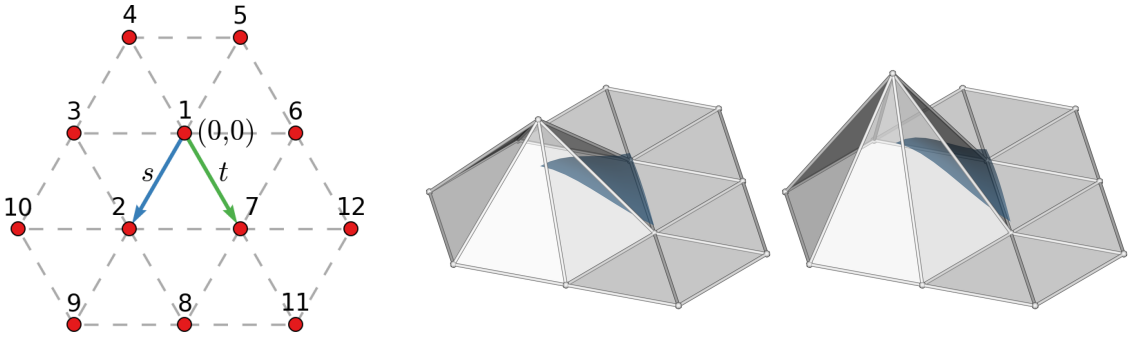
4.1.1 Loop Subdivision Surfaces

Quadratic triangle B-splines are a generalisation of uniform quadratic B-splines to *regular* triangle meshes. A single *ordinary* triangle B-spline patch is defined for 12



(a) (b)

Figure 4.1: (a) One definition for the surface domain of a regular tetrahedron. Each patch domain Ω_i (yellow, blue, green, red) corresponds to a single triangle face (patch) of the same colour. The local patch domains are disjoint and $\Omega = \bigcup_{i=1}^{4} \Omega_i$. Example evaluation points are shown in purple, light blue, and orange. (b) Multiple views of the regular tetrahedron. The control mesh (gray) defines the geometry of the surface and positions of the example evaluation points.



(a) (b)

Figure 4.2: (a) Twelve control vertices (red) with a regular control mesh (gray) define the triangle B-spline patch over the local patch domain axes (blue and green). (b) The geometry of the patch (blue) changes with the positions of the control vertices.

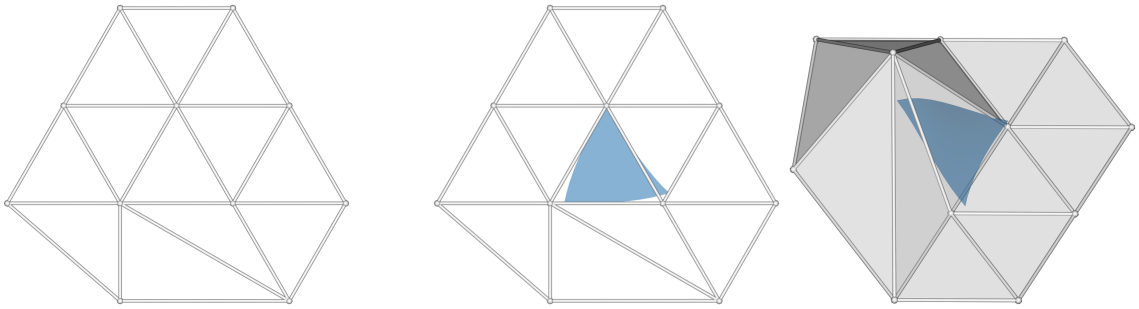
control vertices with regular connectivity (Figure 4.2):

$$\chi\left(\boldsymbol{t}, X\right) = \sum_{i=1}^{12} b_i\left(s,t\right) \boldsymbol{x}_i \tag{4.1}$$

where $\boldsymbol{t} = (s,t)$ is the *local* patch coordinate and each b_i is a quartic polynomial in s and t (§A.1.1). The *local* patch domain is the unit triangle: $\boldsymbol{t} \in \triangle$. ν is defined as the normalised cross-product of the derivatives of χ with respect to s and t. The first and second derivatives of χ are denoted by χ_x and χ_{xy} respectively ($x, y \in \{s, t, X\}$).

For a control mesh with a single *extraordinary* vertex — a vertex with valency not equal to six (Figure 4.3a) — (4.1) *cannot* be applied. Instead, the extraordinary

patch is evaluated by Loop subdivision [93] which generalises the evaluation of triangle B-splines to arbitrary triangle meshes (Figure 4.3b).



(a) (b)

Figure 4.3: (a) A triangle mesh (gray) with a single extraordinary vertex of valency five. (b) The patch (blue) is defined by successive Loop subdivision.

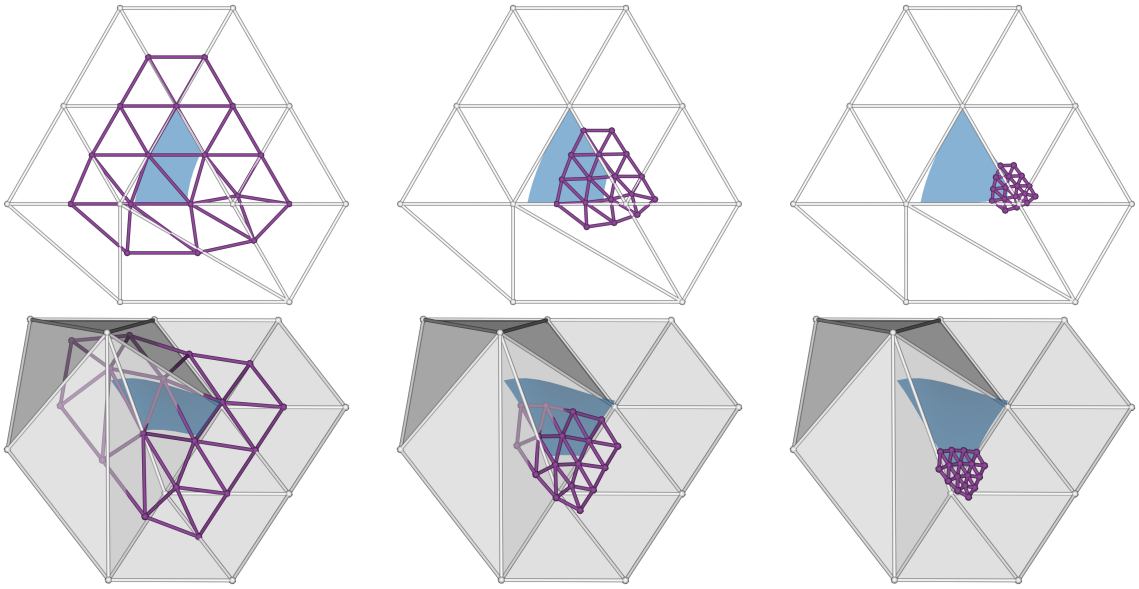
Simply put, Loop subdivision is an algorithm that *refines* a triangle mesh. The refined mesh has more control vertices, and the positions of the refined control vertices are linear functions of the initial control vertex positions [93, 141]. The subdivision weights are defined to maintain first- and second-order parametric and geometric continuity (where possible) so that adjacent surface patches join seamlessly.

Consider Figure 4.4. Loop subdivision of a patch with a single extraordinary vertex of valency $N = 5$ generates a control mesh with *three* ordinary triangle B-spline patches, which are capable of being evaluated using (4.1), and a single extraordinary patch with *identical* topology to the initial mesh (Figure 4.4a). Recursively subdividing the remaining extraordinary component "uncovers" additional ordinary patches so that the extraordinary patch is completely defined (Figures 4.4b and 4.4c).

For patches with multiple extraordinary vertices, one application of Loop subdivision generates child patches with isolated extraordinary vertices so that the above procedure still applies.

Following [141][2], the closed-form for χ for a patch with a single extraordinary vertex is defined as follows. Assume that the local patch domain is defined so that the extraordinary vertex corresponds to $\boldsymbol{t} = (0, 0)$. For an arbitrary patch coordinate $\boldsymbol{t} = (s, t)$, the number of required subdivisions n and (zero-based) child index k are

[2]The variable changes $v \to s$ and $w \to t$ have been made here.



(a) (b) (c)

Figure 4.4: (a) Loop subdivision generates the refined control mesh (purple) from the top-level control mesh (gray). Three ordinary triangle B-spline patches are generated, and the topology of the (unevaluated) extraordinary patch is identical to the top-level. (b), (c) Recursive Loop subdivision of the extraordinary component (purple) enables complete evaluation of the surface patch.

given by:

$$n = \lfloor -\log_2 (s+t) + 1 \rfloor$$
$$k = \begin{cases} 0\colon & 2^{-n} < s \leq 2^{-n+1}, 0 \leq t \leq 2^{-n+1} - s \\ 1\colon & 0 \leq s \leq 2^{-n}, 2^{-n} - s \leq t \leq 2^{-n} \\ 2\colon & 0 \leq s \leq 2^{-n}, 2^{-n} < t \leq 2^{-n+1} - s \end{cases}$$

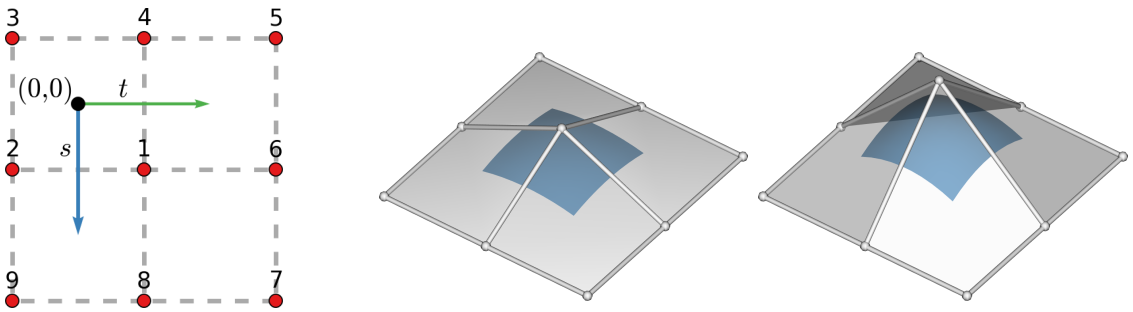
With $X \in \mathbb{R}^{3\times(N+6)}$ denoting the matrix of positions of the $N+6$ patch control vertices, χ is then given by:

$$\chi(\boldsymbol{t}, X) = X\left(P_k \bar{A} A^{n-1}\right)^\top \boldsymbol{b}\left(t_{k,n}(\boldsymbol{t})\right) \tag{4.2}$$

where $\boldsymbol{b}$ is the vector of triangle B-spline basis functions and $t_{k,n}$ transforms $\boldsymbol{t}$ to the local *subdivided* patch domain for child k after n subdivisions:

$$t_{0,n}(\boldsymbol{t}) = (2^n s - 1, 2^n t), \quad t_{1,n}(\boldsymbol{t}) = (1 - 2^n s, 1 - 2^n t), \quad t_{2,n}(\boldsymbol{t}) = (2^n s, 2^n t - 1)$$

and, with reference to [141], $P_k \in \mathbb{R}^{12\times(N+12)}$ is the *"picking"* matrix which selects the k^{th} ordinary triangle B-spline patch vertices after n subdivisions, $A \in \mathbb{R}^{(N+6)\times(N+6)}$ is the *"extended"* subdivision matrix which subdivides the extraordinary component



(a) (b)

Figure 4.5: (a) Nine control vertices with a regular control mesh (gray) define the biquadratic B-spline patch over the local patch domain axes (blue and green). (b) The geometry of the patch (blue) changes with the positions of the control vertices.

(§A.1.2), and $\bar{A} \in \mathbb{R}^{(N+12)\times(N+6)}$ is the *"bigger"* subdivision matrix which subdivides the extraordinary component *and* generates the patch vertices which define the three "uncovered" ordinary triangle B-spline patches. With reference to Figure 4.4, the subdivided control meshes (purple) are $X\left(\bar{A}A^0\right)^\top$, $X\left(\bar{A}A^1\right)^\top$, and $X\left(\bar{A}A^2\right)^\top$.

Efficient evaluation of (4.2) uses decompositions of A to evaluate A^n. For $N > 3$, A is diagonalisable so that $A = V\Lambda V^{-1}$, where Λ is the diagonal matrix of eigenvalues and V is the matrix of eigenvectors. For $N = 3$, A is *not* diagonalisable because the columns of V are not independent. Instead, the Jordan normal form is used, where Λ is an upper triangular matrix with a single non-zero off-diagonal of 1. The complete derivation of the eigenstructure is supplied in [141], but details regarding the necessary Fourier analysis of A are absent. For completeness, the omitted steps have been derived independently and are supplied in §A.1.3.

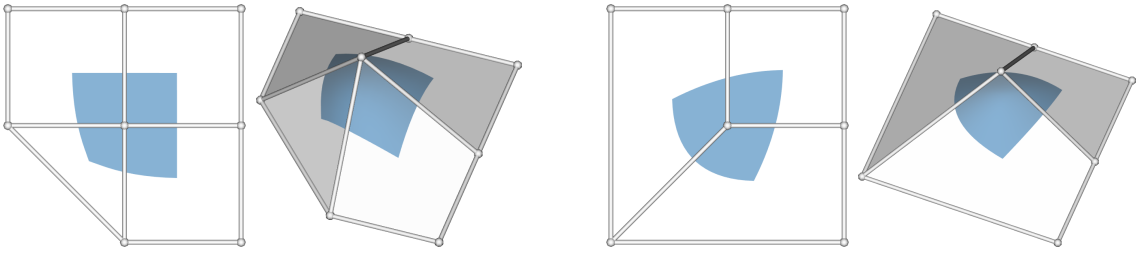
4.1.2 Doo-Sabin Subdivision Surfaces

A biquadratic B-spline is a generalisation of uniform quadratic B-splines to *regular* quadrilateral meshes [124]. A single ordinary biquadratic B-spline patch is defined for nine control vertices with regular connectivity (Figure 4.5):

$$\chi\left(\boldsymbol{t}, X\right) = \sum_{i=1}^{9} b_i\left(s,t\right)\boldsymbol{x}_i \tag{4.3}$$

where $\boldsymbol{t} = (s,t)$ is the local patch coordinate and each b_i is a quartic polynomial in s and t (§A.2.1). The local patch domain is the unit square: $\boldsymbol{t} \in \square = \{(s,t) : s,t \in [0,1]\}$. As before, ν is defined as the normalised cross-product of the derivatives of χ with respect to s and t.

Similar to §4.1.1, biquadratic B-splines are *not* defined for patches with non-quadrilateral faces *or* a centre vertex with valency not equal to four (Figure 4.6).



(a) (b)

Figure 4.6: (a) A mesh (gray) with a single extraordinary non-quadrilateral (triangle) face and the Doo-Sabin patch defined by successive subdivision (blue). (b) A single extraordinary vertex with valency three and the Doo-Sabin patch.

Consider Doo-Sabin subdivision applied to a patch with an extraordinary face with $N = 5$ sides *and* a centre vertex of valency three (Figure 4.7). One application of Doo-Sabin subdivision generates a control mesh with three extraordinary biquadratic B-spline patches (Figure 4.7a). Two patches have a single non-quadrilateral (triangle) face; the third patch has both triangle and pentagon faces. The second application of Doo-Sabin subdivision (Figure 4.7b) generates *four* child patches for each extraordinary patch, eight of which are ordinary biquadratic B-spline patches and are evaluable using (4.3). The four extraordinary patches (Figure 4.7c) only contain a single non-quadrilateral face. Subsequent subdivision of the extraordinary patches generates additional ordinary patches and the complete patch is evaluable (Figures 4.7d, 4.7e, and 4.7f). Importantly, comparing Figures 4.7c and 4.7e, the topology of the extraordinary patches remains constant throughout.

Exact evaluation of Doo-Sabin surfaces is performed almost identically to Loop [141] and Catmull-Clark[3] subdivision surfaces [142]. The definitions for n, k, and $t_{k,n}$ (§A.2.3) are analogous to those defined in [142], and the subdivision matrices (§A.2.2) are defined by the subdivision weights in [47].

In the experiments that follow, only Doo-Sabin subdivision surfaces with control meshes with all vertices of valency four are used. This ensures all patches are quadrilateral, enabling a consistent parameterisation for all patches which simplifies the model surface implementation and model coordinate update algorithm (§4.2).

[3]Catmull-Clark subdivision generalises uniform *bicubic* B-spline patches, which are defined on a regular quadrilateral mesh of 16 control vertices.

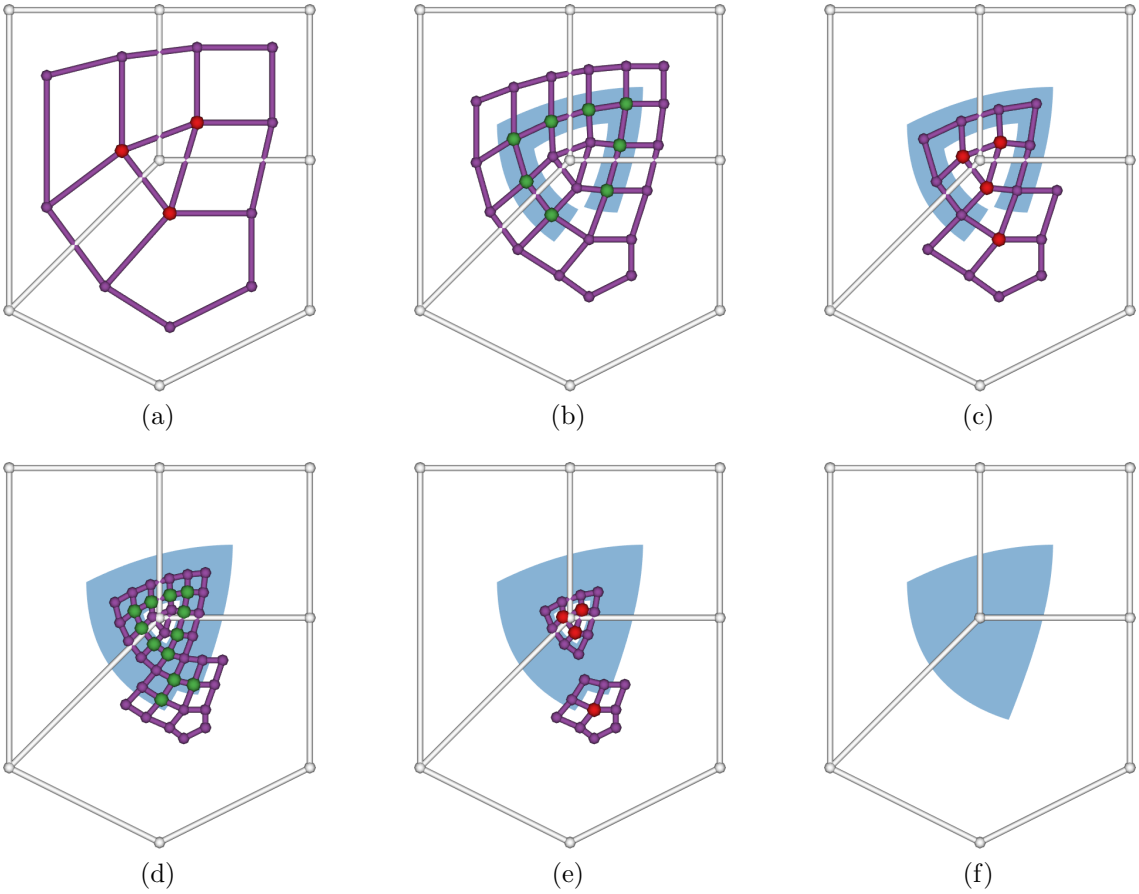


Figure 4.7: (a) A mesh (gray) with an extraordinary non-quadrilateral (pentagon) face *and* a centre vertex of valency three. The subdivided mesh (purple) has three extraordinary patches (red). (a)–(f) Recursive subdivision of the extraordinary patches generates ordinary biquadratic B-spline patches (green) and enables complete evaluation of the surface patch.

4.1.3 Degenerate Parameterisation Properties

In 2D, continuous optimisation of model contour control points and correspondences requires well-defined first and second derivatives for χ with respect to t and X (§3.3.4.2). In `LM`, first derivatives are necessary to evaluate the Jacobian entries for residuals using χ. Similarly, second derivatives are necessary to evaluate the Jacobian entries, via the chain rule, for residuals using ν. In 3D, identical requirements for the model surface apply — it must have well-defined first and second derivatives for χ with respect to $\boldsymbol{t} = (s, t)$ and X.

For ordinary and extraordinary Loop and Doo-Sabin surface patches, χ is linear with respect to X, and first and second derivatives with respect to X are therefore well-defined and straightforward to evaluate. With respect to $\boldsymbol{t}$, derivatives for

ordinary triangle B-spline (4.1) and biquadratic B-spline (4.3) patches are well-defined and evaluable by replacing each b_i with its first or second derivative, $\partial_x b_i$ or $\partial^2_{xy} b_i$, respectively ($x, y \in \{s, t\}$).

For extraordinary patches, first (or second) derivatives can be evaluated by replacing b_i with $\partial_x b_i$ ($\partial^2_{xy} b_i$) in (4.2) *and* multiplying by $\partial_x t_{k,n}$ ($\partial_x t_{k,n} \partial_y t_{k,n}$) [141, 142]. One (minor) practical drawback of this method is that precision of the derivatives decreases for $\boldsymbol{t} \to \mathbf{0}$. If $\boldsymbol{t} \to \mathbf{0}$, then $n \to \infty$, and $X(A^{n-1})^\top \to X\tilde{\boldsymbol{v}}_1 \mathbf{1}^\top$, where $\tilde{\boldsymbol{v}}_1$ is the left eigenvector of A corresponding to the eigenvalue 1, and $\mathbf{1}^\top$ is a row vector of ones. Therefore, the precision of the derivatives is limited because, for large n, differences between columns of $(A^{n-1})^\top$ are only in the lower bits of the floating point entries.

Importantly, while derivatives of (4.2) are straightforward to evaluate, whether or not they are finite and bounded for $\boldsymbol{t} \to \mathbf{0}$ is not obvious by inspection.

Unfortunately, for extraordinary Loop patches with a vertex of valency $N < 6$, the first derivatives vanish: $\lim_{\boldsymbol{t} \to \mathbf{0}} \|\chi_x(\boldsymbol{t}, X)\| = 0$ (§A.1.7). This is undesirable for model surface fitting because it introduces saddle points which are detrimental to derivative based continuous optimisation over $\boldsymbol{t}$. Furthermore, for $N > 6$, the first derivatives are unbounded[4]: $\lim_{\boldsymbol{t} \to \mathbf{0}} \|\chi_x(\boldsymbol{t}, X)\| \to \infty$. Finally, for $N \notin \{3, 6\}$, second derivatives are unbounded (§A.1.7). This is undesirable because it prevents simple chain rule evaluation of model surface normal derivatives. Furthermore, it prevents the direct use of surface regularisers based on integrals of second derivatives, such as the thin-plate energy [63].

For extraordinary Doo-Sabin patches, first derivatives are finite, but second derivatives are unbounded for $N > 4$ (§A.2.4).

The simplest solution to overcome the degenerate behaviour of extraordinary patches is to limit the number of subdivisions performed during evaluation of points and derivatives. This is exactly equivalent to disallowing local patch coordinates close to $\mathbf{0}$ so that vanishing and unbounded first and second derivatives are avoided entirely. This approach is straightforward to implement but has the disadvantage that it inherently limits the model surface domain and might hinder model surface fitting.

Another solution is to seek an alternative to the "natural" parameterisation of the subdivision patches which does not suffer from degenerate derivatives. By way of example, consider the 2D parametric curve $(x, y) = (\sqrt{s}, \sin(\sqrt{s}))$, with $s \in \mathbb{R}_{\geq 0}$. Its first derivative with respect to s is $\left(\frac{1}{2\sqrt{s}}, \frac{1}{2\sqrt{s}} \cos(\sqrt{s})\right)$ which is unbounded as $s \to 0$. However, with the reparameterisation $s = t^2$, the curve becomes $(x, y) = (t, \sin(t))$ and its first derivative with respect to t is $(1, \cos(t))$, which *is* finite at $t = 0$. Similarly

[4]Similar behaviour has been noted for Catmull-Clark surfaces [20].

motivated approaches have been pursued for Catmull-Clark subdivision patches, where the characteristic map (§A.1.6) is used to construct the reparameterisation function, but surface point and derivative evaluation is subsequently expensive [20].

A third solution is to replace an extraordinary patch, after some fixed number of subdivisions, with a different surface model entirely. For Loop patches, a quartic triangle Bezier approximation has been developed (§A.1.9). The triangle Bezier patch functions are linear in the top-level control vertices, non-linear in $\boldsymbol{t} = (s, t)$, and maintain geometric and tangent plane continuity to adjacent ordinary triangle B-spline patches. However, very minor tangent plane discontinuities do occur between Bezier patches but these are negligible in practice.

In the experiments that follow, standard Doo-Sabin surfaces were used with an upper limit of 10 subdivisions. For Loop surfaces, the surrogate Bezier patches were used in place of extraordinary patches.

4.2 Model Coordinate Updates

Application of `LM` for model surface fitting has two requirements: evaluation of the system Jacobian and a suitable procedure for applying updates (§3.3.4.2). The primary focus of this chapter so far has been the construction of model surfaces which extend uniform quadratic B-spline model contours. In particular, the degenerate properties of first and second derivatives of subdivision surfaces have been detailed and addressed so that the first requirement for `LM` is satisfied. The purpose of this section is to address the second requirement and generalise "correspondence addition" from uniform quadratic B-splines (3.25) to subdivision surfaces.

Triangle Mesh Surfaces Consider a Loop subdivision surface defined by a triangle mesh (Figure 4.8a). Let $\boldsymbol{u} \in \Omega$ denote a single model coordinate and let $(i, \boldsymbol{t}) = \Psi(\boldsymbol{u})$, where i is a (one-based) patch index and $\boldsymbol{t} = (s, t) \in \triangle$ is the local patch coordinate in the unit triangle (§4.1). Also, let $\delta\boldsymbol{t} \in \mathbb{R}^2$ denote a local model coordinate update proposed by `LM`. If $\boldsymbol{t} + \delta\boldsymbol{t} \in \triangle$, then the updated model coordinate is simply $\Psi^{-1}(i, \boldsymbol{t} + \delta\boldsymbol{t})$. I.e. the model coordinate remains in the same patch. This is identical to adding a small update to a model coordinate on a uniform quadratic B-spline such that it remains in the same segment.

For $\boldsymbol{t} + \delta\boldsymbol{t} \notin \triangle$ it is necessary to transition between patches, which is analogous to transitioning between segments on closed uniform quadratic B-splines. The later is trivially handled using fmod (3.25) because the model contour domain Ω is defined so

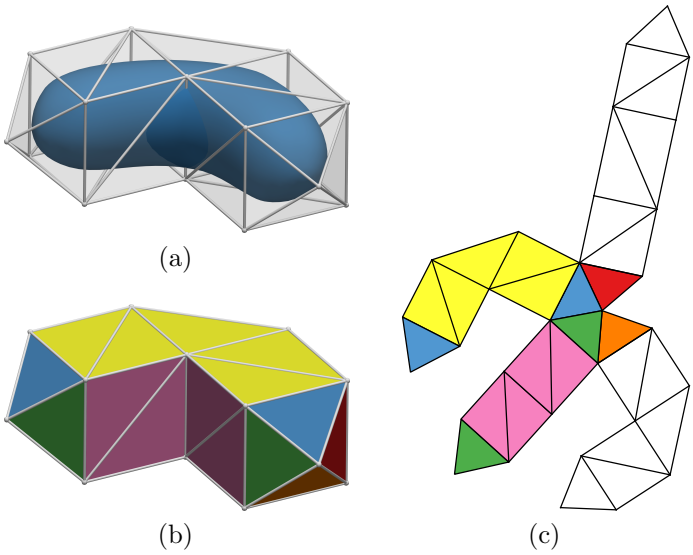


Figure 4.8: (a) An example Loop subdivision surface (blue) defined by its control mesh (gray). (b), (c) The colour-coded control mesh and its unwrapping (occluded faces are white).

that all pairs of adjacent segments, except the start and end, have adjacent segment domains Ω_i of unit length. Unfortunately such a simple function is not available on a triangle mesh. This is because defining the model surface domain Ω so that adjacent patches i and j have connected Ω_i and Ω_j is not possible in general, even when Ω_i and Ω_j are *not* restricted to right-angled triangles or aligned on the same axes (Figures 4.8b, 4.8c).

The algorithm to perform patch traversal is similar to [28] and has been used in [150]. Starting with $(i, \boldsymbol{t})$ and $\delta\boldsymbol{t}$, the maximum scale $0 \leq \sigma < 1$ is found so that $\boldsymbol{t} + \sigma\delta\boldsymbol{t}$, denoted by $\boldsymbol{t}_i^{\triangle}$, is on a *boundary* of $\triangle$. Next, let j denote the adjacent patch index and $\boldsymbol{t}_j^{\triangle}$ denote the local patch coordinate in patch j which evaluates to the same model surface position, i.e. $\chi^i\left(\boldsymbol{t}_i^{\triangle}, X\right) = \chi^j\left(\boldsymbol{t}_j^{\triangle}, X\right)$. Finally, let $\delta\boldsymbol{t}_i^{\triangle} = (1-\sigma)\,\delta\boldsymbol{t}$ denote the "remaining" update vector in patch i. The local patch coordinate update in patch j, denoted by $\delta\boldsymbol{t}_j^{\triangle}$, is found by minimising the squared Euclidean norm of the tangent vectors evaluated on both sides of the patch boundary[5]:

$$\delta\boldsymbol{t}_j^{\triangle} = \arg\min_{\boldsymbol{\delta}} \left\| \left[\chi_s^i\left(\boldsymbol{t}_i^{\triangle}, X\right) \quad \chi_t^i\left(\boldsymbol{t}_i^{\triangle}, X\right)\right] \delta\boldsymbol{t}_i^{\triangle} - \left[\chi_s^j\left(\boldsymbol{t}_j^{\triangle}, X\right) \quad \chi_t^j\left(\boldsymbol{t}_j^{\triangle}, X\right)\right] \boldsymbol{\delta} \right\|^2$$

where χ_x^i is the first derivative of χ^i with respect to $x \in \{s, t\}$. The closed-form for

[5]The minimum squared Euclidean norm of the tangent vectors is only non-zero at boundaries between extraordinary patches as a result of the (minor) tangent plane discontinuities of the approximate Loop patches (§A.1.9).

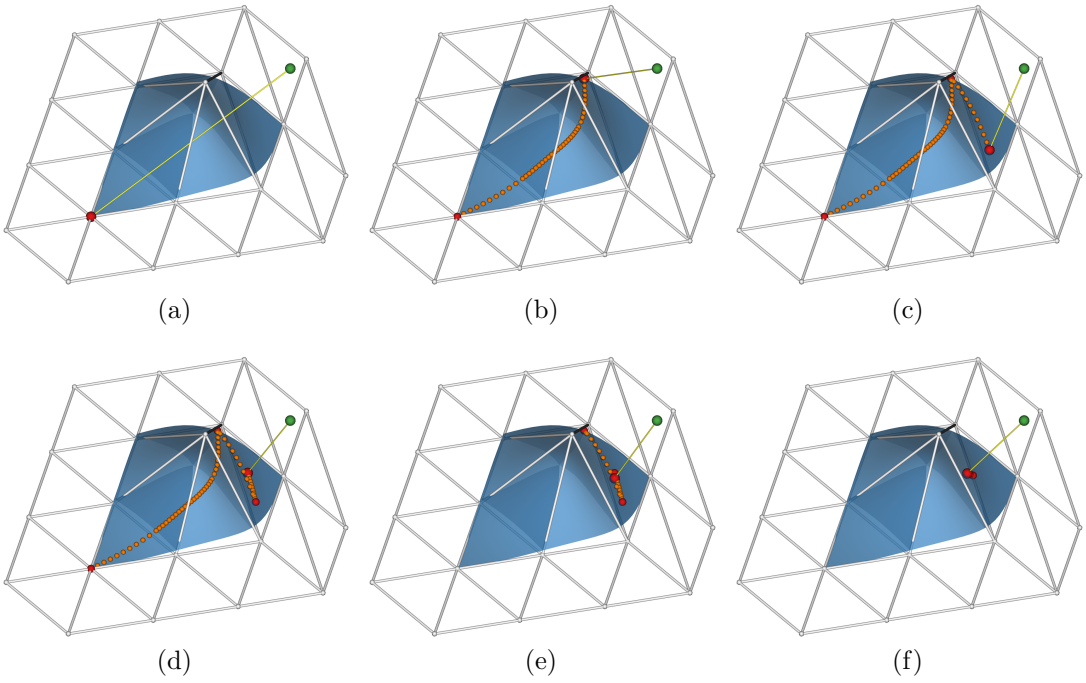


Figure 4.9: Example `LM` iterations minimising the closest point problem of (4.4). At each iteration, the error (yellow) between the model point (red) and reference point (green) is reduced. The last three model coordinate positions (small, red) and transitions (orange) are shown when available. (a) Initialisation. (b)–(e) Results of five `LM` iterations. (f) Result after 10 `LM` iterations.

$\delta \boldsymbol{t}_j^{\triangle}$ is:

$$\delta \boldsymbol{t}_j^{\triangle} = \left[\chi_s^j \left(\boldsymbol{t}_j^{\triangle}, X \right) \quad \chi_t^j \left(\boldsymbol{t}_j^{\triangle}, X \right) \right]^{\dagger} \left[\chi_s^i \left(\boldsymbol{t}_i^{\triangle}, X \right) \quad \chi_t^i \left(\boldsymbol{t}_i^{\triangle}, X \right) \right] \delta \boldsymbol{t}_i^{\triangle}$$

where $A^{\dagger}$ is the Moore-Penrose inverse of a matrix A [131]. Setting $\boldsymbol{t} = \boldsymbol{t}_j^{\triangle}$ and $\delta \boldsymbol{t} = \delta \boldsymbol{t}_j^{\triangle}$, the algorithm is repeated until $\boldsymbol{t} + \delta \boldsymbol{t} \in \triangle$, a patch boundary is met, or until a patch is revisited. The final model coordinate is recovered using Ψ^{-1}. Importantly, this algorithm is applicable to any surface model with a triangle patch structure; it is *not* restricted solely to Loop subdivision surfaces.

The use of "correspondence addition" with `LM` is demonstrated in Figure 4.9 for the problem of finding the model coordinate $\boldsymbol{u}$ of the point on a Loop subdivision surface which is closest to a reference point $\boldsymbol{y} \in \mathbb{R}^3$:

$$\min_{\boldsymbol{u}} \left\| \boldsymbol{y} - \chi \left(\boldsymbol{u}, X \right) \right\|^2 \tag{4.4}$$

Although the initialisation for $\boldsymbol{u}$ is poor (Figure 4.9a), the first iteration of `LM` "pulls" the model coordinate across the surface towards the reference (Figure 4.9b). Importantly and as expected, the transitions between patches are smooth. Also, although

the update is severely distorted away from its initial heading by the extraordinary vertex, the updated model coordinate is still accepted because the distance to the reference is reduced. It is quickly refined in subsequent iterations (Figures 4.9c–4.9e), and the result after 10 iterations is close to the global minimum (Figure 4.9f).

Doo-Sabin Subdivision Surfaces A Doo-Sabin subdivision surface with all vertices of valency four has a regular quadrilateral patch structure. In this case, the algorithm for triangle mesh surfaces can be used by replacing $\triangle$ with $\square$ for the domain of the local patch coordinates.

4.3 Additional Framework Component Details

The definitions for different model surfaces (§4.1) and the algorithm for performing model coordinate updates (§4.2) enable 3D model-based segmentation to be formulated and optimised almost identically to in 2D. However, definitions for boundary candidate selection models and surface regularisers are still required. Describing these additional components and providing further implementation details, in preparation for the applications that follow, is the purpose of this section.

4.3.1 Boundary Candidate Selection

Given the definition for the model surface, extending single-frame fitting and boundary candidate selection algorithms from 2D to 3D is straightforward.

Let $X \in \mathbb{R}^{3\times N_X}$ denote the matrix of control vertices which define the model surface. Identical to §3.1.3, assume that N_U points on the surface are sampled at coordinates specified by the matrix U, where each $\boldsymbol{u}_i \in \Omega$. Let $\phi \in \mathbb{R}^{3\times N_\phi}$ denote the matrix of boundary candidate positions.

Perpendicular boundary candidate selection is then defined analogously to (3.1). For each sampled coordinate on the model surface, indexed by i, its associated boundary candidate index l_i is given by:

$$l_i = \underset{l\in\Upsilon(\boldsymbol{u}_i,X)}{\arg\min} \left\|\boldsymbol{\phi}_l - \chi\left(\boldsymbol{u}_i, X\right)\right\|^2$$

where, as before, Υ is a function which returns the set of indices of boundary candidates which are perpendicular to the model point at coordinate $\boldsymbol{u}_i$. Unconstrained boundary candidate selection is defined identically but *without* the constraint that $l \in \Upsilon\left(\boldsymbol{u}_i, X\right)$.

In §3.2.6, pairwise boundary candidate constraints were introduced to encourage coherent selection of boundary candidates in 2D. The extension to 3D is straightforward.

Let $\mathcal{T}_U$ denote the set of edges which define a mesh over the samples U on the model surface. The energy over boundary candidate indices with added pairwise constraints is:

$$E\left(\boldsymbol{l} \mid U, X\right) = \sum_{i=1}^{N_U} \left\| \boldsymbol{\phi}_{l_i} - \chi\left(\boldsymbol{u}_i, X\right) \right\|^2 + \lambda_{\mathrm{P}} \underbrace{\sum_{(i,j)\in\mathcal{T}_U} \left\| \boldsymbol{\phi}_{l_i} - \boldsymbol{\phi}_{l_j} \right\|^2}_{E_{\mathrm{P}}(\boldsymbol{l})} \tag{4.5}$$

which is a first-order `MRF` with pairwise potentials similar to (3.21). Therefore, the algorithm given in §3.3.4.1 is used to approximately minimise (4.5) — Prim's algorithm and `MIN-SUM` initialise $\boldsymbol{l}$, and it is subsequently refined using `QPBO` (§3.3.4.1).

Finally, if boundary candidate normals are available and denoted by the matrix $\eta \in \mathbb{R}^{3\times N_\phi}$, then the oriented surface fit is defined analogously to (3.26). All unary potentials for each boundary candidate selection algorithm are replaced accordingly. Perpendicular boundary candidate selection is given by:

$$l_i = \underset{l\in\Upsilon(\boldsymbol{u}_i, X)}{\arg\min} \left\{ \left\| \boldsymbol{\phi}_l - \chi\left(\boldsymbol{u}_i, X\right) \right\|^2 + \lambda_\eta \left\| \boldsymbol{\eta}_l - \nu\left(\boldsymbol{u}_i, X\right) \right\|^2 \right\} \tag{4.6}$$

with unconstrained boundary candidate selection defined similarly. Likewise, (4.5) becomes:

$$E\left(\boldsymbol{l} \mid U, X\right) = \sum_{i=1}^{N_U} \left\{ \left\| \boldsymbol{\phi}_{l_i} - \chi\left(\boldsymbol{u}_i, X\right) \right\|^2 + \lambda_\eta \left\| \boldsymbol{\eta}_{l_i} - \nu\left(\boldsymbol{u}_i, X\right) \right\|^2 \right\} + \lambda_{\mathrm{P}} E_{\mathrm{P}}\left(\boldsymbol{l}\right) \tag{4.7}$$

which is solved identically to before.

4.3.2 Regularisers

Similar to uniform quadratic B-splines, regularisers over subdivision surfaces are straightforward to specify. The most straightforward regulariser for the model surface encourages adjacent control vertices to be as close as possible to each other:

$$E_{\textsc{Close}}\left(X\right) = \sum_{(i,j)\in\mathcal{T}} \left\| \boldsymbol{x}_i - \boldsymbol{x}_j \right\|^2 \tag{4.8}$$

which resembles (3.17).

For Loop subdivision surfaces with approximate patches, the first and second derivatives are bounded and physically motivated regularisers such as the thin-plate energy can be used. For the patch indexed by i, the isotropic thin-plate energy is given by [63]:

$$E^i_{\mathrm{TP}}\left(X\right) = \int_{\boldsymbol{t}\in\triangle} \left\| \chi^i_{xx}\left(\boldsymbol{t}, X\right) \right\|^2 + 2\left\| \chi^i_{xy}\left(\boldsymbol{t}, X\right) \right\|^2 + \left\| \chi^i_{yy}\left(\boldsymbol{t}, X\right) \right\|^2 \, \mathrm{d}\boldsymbol{t} \tag{4.9}$$

where the derivatives of χ^i are functions of $\boldsymbol{t}$, but the derivatives are with respect to an orthogonal coordinate system which ensures an isotropic thin-plate definition [28]. The total thin-plate energy for a surface is the sum over all patches. Crucially, the derivatives of χ^i are linear in X so that the evaluated thin-plate integral is quadratic in X and `LM` remains suitable for local continuous optimisation[6].

For Doo-Sabin subdivision surfaces, thin-plate regularisation is not applicable, without modification[7], since second derivatives are unbounded for $N > 4$.

4.3.3 Model Surface Implementation

Loop Subdivision Surfaces An arbitrary Loop subdivision surface with mesh $\mathcal{T}$ can contain many different patch *types*, each with one or more extraordinary vertices that is approximated by one or more triangle Bezier patches. Therefore, the definitions for χ, ν, each E^i_{TP}, and the associated derivatives, can be complex. To simplify the general use of these surfaces for local continuous optimisation with `LM`, and the `CERES` [2] framework in particular, a Python 2.7 program was written which, given an input mesh $\mathcal{T}$, generates bespoke and optimised C code for χ, ν, each E^i_{TP}, and the associated derivatives. All functions are callable through a consistent C++ interface and generated Python extension module.

To start, the unique top-level patch types in an input mesh $\mathcal{T}$ are discovered. Next, a *symbolic* formulation for each (potentially piecewise) patch position function is constructed using `Sympy`[8]. For the ordinary B-spline patch type, the patch function is given by (4.1). For an extraordinary patch type with a single extraordinary vertex, the patch function *can* be given immediately by its Bezier replacement (A.17), where the Bezier coefficients are evaluated symbolically according to §A.1.9 and the substitution $r = 1 - s - t$ is made. Alternatively, the extraordinary patch can be subdivided (symbolically) for a fixed number of iterations before the extraordinary component is replaced. In this case, the patch function is piecewise, consisting of ordinary triangle B-spline patches and the replacement triangle Bezier patch. Extraordinary patch types with multiple extraordinary vertices are handled similarly, requiring at least one subdivision before replacing the extraordinary components.

Given the symbolic formulation of each position function for each patch type, formulations for the first and second derivatives are generated and the thin-plate

[6] A demonstration of evaluating the quadratic form for an ordinary triangle B-spline patch is given in §A.1.4.

[7] Application of thin-plate regularisation to Catmull-Clark surfaces is detailed in [63].

[8] `http://www.sympy.org`

quadratic form (§A.1.4) is evaluated. All constituent patch functions are then optimised by local common subexpression elimination [105] and outputted as C code. Finally, the parent evaluation functions — which handle the piecewise definitions of each patch function and the mapping of patch indices to patch types — are outputted.

The advantages of enumerating patch functions for a given $\mathcal{T}$ are significant. First, subdivision is *not* performed at run-time so that evaluation of χ, ν, E^i_{TP}, and the associated derivatives, is very fast. Second, since the subdivided patch functions — which are of the form (4.2) — are evaluated symbolically and *exactly* prior to code generation, evaluation of first and second derivatives does *not* suffer any loss of precision if the number of subdivisions at extraordinary patches is increased (cf. §4.1.3).

Doo-Sabin Subdivision Surfaces Doo-Sabin subdivision surfaces are comparatively simpler to implement because only control meshes with vertices of valency four are considered, and no approximate patches or thin-plate quadratic forms are required. A C++ implementation has been developed which performs evaluation by recursive subdivision (4.2), with the definitions for n, k, $t_{k,n}$, and the subdivision matrices, given in §A.2.3. To improve run-time performance, the recursively generated subdivision matrices are memoised.

4.3.4 Data-to-Model and Robust Surface Fitting

Up to this point, the algorithms described for contour and surface fitting have been limited to "model-to-data" approaches, where boundary candidate indices $\boldsymbol{l}$ are set *given* model coordinates U. Perpendicular (§3.1.3) and unconstrained (§3.2.4) boundary candidate selection, and pairwise constraints (§3.2.6) and robust formulations (§3.2.7), are examples and extensions of this approach. The opposite, "data-to-model", initialises model coordinates U *given* boundary candidate indices $\boldsymbol{l}$. That is, a model coordinate is introduced for *every* selected or available boundary candidate.

The advantage of "model-to-data" is that anatomical consistency (§3.2.5) and robustness to missing or incorrect boundary candidates are both straightforward to formulate. However, delineation accuracy is inherently limited because a small subset of boundary candidates are selected, *and* the selection depends on the current or initial distribution of points on the model contour or surface. For applications where the vast majority of boundary candidates are valid, or a selection of valid boundary candidates can be made *independent* of U, the "data-to-model" approach can achieve

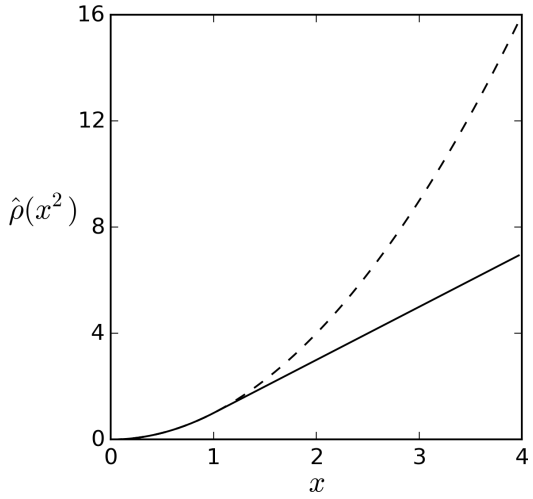


Figure 4.10: The Huber loss function (solid) and identity function (dashed).

superior delineation accuracy simply because it makes use of more available boundary candidates.

A complication of "data-to-model" is that the position and orientation errors in the continuous energy must be made robust *if* invalid boundary candidates are selected or present. So far, this has not been necessary because in "model-to-data" it is assumed that invalid boundary candidates have been excluded from the selection. Fortunately, robustification in non-linear least squares problems has been well-studied in the field of *bundle adjustment* [151] and is achieved by introducing a *loss function* ρ which reduces the influence of large squared errors. Position errors are then replaced:

$$\left\| \boldsymbol{\phi}_{l_i} - \chi\left(\boldsymbol{u}_i, X\right) \right\|^2 \rightarrow \rho\left(\left\| \boldsymbol{\phi}_{l_i} - \chi\left(\boldsymbol{u}_i, X\right) \right\|^2 \right)$$

with a similar replacement made for orientation errors. Although using ρ means that the position and orientation errors are *not* squared residuals, local continuous optimisation can still be performed using `LM`, but the residuals and Jacobian blocks must be scaled according to [151]. (This is handled automatically in `CERES`.)

Multiple choices for loss functions exist [2], but for simplicity, the Huber loss function [73] (Figure 4.10) is used in the applications that follow. It is parameterised by a scale δ and defined as in [2]:

$$\rho\left(x, \delta\right) = \delta^2 \hat{\rho}\left(\frac{x}{\delta^2}\right), \quad \hat{\rho}\left(x\right) = \begin{cases} x, & x \leq 1 \\ 2\sqrt{x} - 1, & x > 1 \end{cases}$$

	Applications		
	Fetal skull	Fetal face	Left ventricle
Model Surface	Doo-Sabin	Loop	Loop
Regulariser	"As close as possible"	Thin-plate	Thin-plate
Fitting Error	Position & orientation	Position	Position
Boundary Candidate Selection	Robust & pairwise constraints	Unconstrained "data-to-model"	Perpendicular & unconstrained "data-to-model"

Table 4.1: Framework components used for each 3D model-based segmentation application.

4.4 Applications

The purpose of this section is to demonstrate that, given the definitions for model surfaces (§4.1), model coordinate updates (§4.2) and additional framework components (§4.3), 3D single-frame model-based segmentation is trivial to formulate, and, the optimisation algorithms developed for 2D+t boundary candidate selection and model contour fitting are immediately applicable.

First, 3D model-based segmentation of the skull from fetal ultrasound images is presented. This work was developed to facilitate brain analysis from fetal ultrasound images and is a crucial preprocessing step in a pipeline for predicting neurodevelopmental age [111]. Second, 3D model-based segmentation of the face from fetal ultrasound images is demonstrated. This work was developed to facilitate parametric analysis of the facial structure of fetuses. Third, 3D model-based segmentation of the left ventricle from echocardiography images is described. Application on a challenge dataset demonstrates that the segmentation accuracy is extremely competitive, despite the framework being generic and *not* modelling some application-specific segmentation minutiae. The different model surfaces, regularisers, fitting errors, and boundary candidate selection models for each application are summarised in Table 4.1.

4.4.1 3D Cranial Parameterisation

The purpose of 3D ultrasound fetal brain image analysis is to investigate the development of intracranial structures using information from different subjects at different stages of growth. Co-localisation of the 3D images is therefore a necessary preprocessing step so that, given a brain region of interest, image information can be consistently

sampled for each subject and image. The typical approach in general neuroimage processing is skull stripping followed by group or pairwise registration [147]. For fetal ultrasound images this is made difficult by the presence of speckle, acquisition artefacts, and variability of occlusions between subjects. As an alternative, a *cranial parameterisation* has been proposed and validated which achieves co-localisation of the brain surface by fitting a parametric surface model of the fetal skull to the inner skull boundary in each image. This section describes the cranial model surface and fitting algorithm introduced in [109] and applied in [110, 111]. In particular, it is described in the context of the 3D model-based segmentation components detailed so far.

With reference to [109], a Doo-Sabin subdivision surface was used for the cranial model surface (Figure 4.11a) and the 96 control vertices and 98 faces were colour-annotated to facilitate *consistent* manual initialisation between subjects and images. The initial surface control vertices, denoted by $X^0 \in \mathbb{R}^{3\times N_X}$, were set by the user rigidly aligning the default skull surface to the imaged brain in a multi-view graphical user interface (Figures 4.11b and 4.11c)[9]. Boundary candidate positions $\phi \in \mathbb{R}^{3\times N_\phi}$ and normals $\eta \in \mathbb{R}^{3\times N_\phi}$ were generated for each 3D ultrasound image using an isotropic log-Gabor filter with feature asymmetry[10] and non-maximum suppression (§3.4.2).

Let X, U, and $\boldsymbol{l}$ be defined as in §4.3.1. Using the oriented surface fit with pairwise constraints (4.7) in conjunction with "as close as possible" surface regularisation (4.8), the complete energy was *initially* defined as:

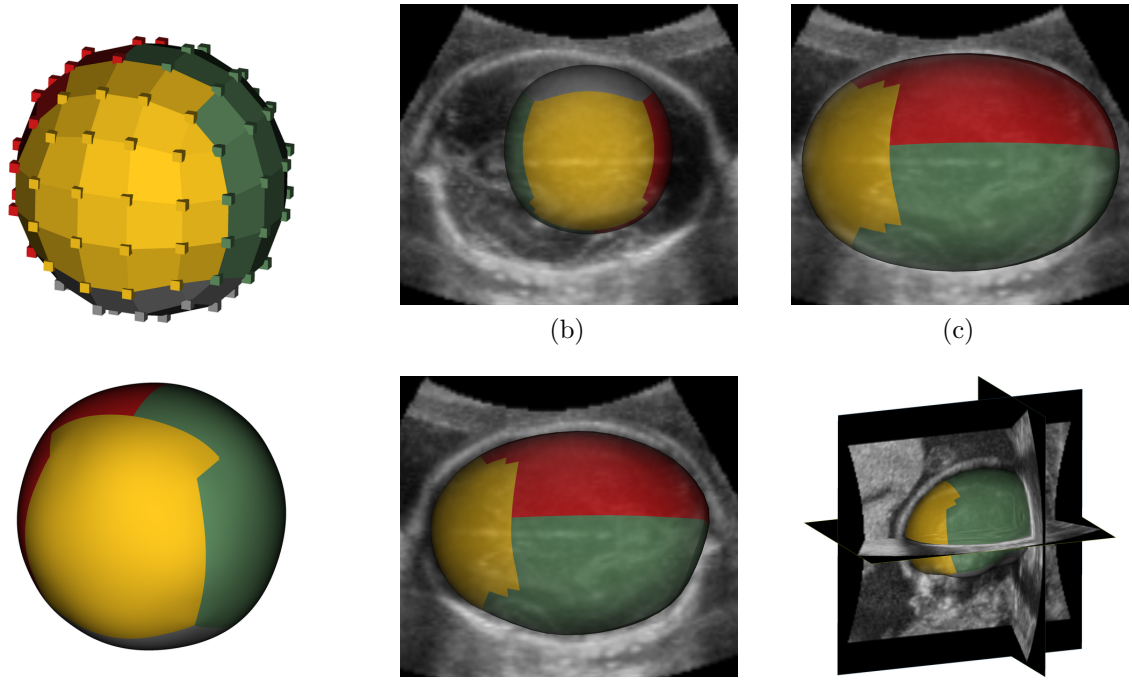
$$\begin{aligned} E\left(X,U,\boldsymbol{l}\right) = & \underbrace{\sum_{i=1}^{N_U}\left\{\left\|\boldsymbol{\phi}_{l_i}-\chi\left(\boldsymbol{u}_i,X\right)\right\|^2+\lambda_\eta\left\|\boldsymbol{\eta}_{l_i}-\nu\left(\boldsymbol{u}_i,X\right)\right\|^2\right\}}_{E_{\text{Fit}}(X,U,\boldsymbol{l})} \\ & +\lambda_{\text{P}}E_{\text{P}}\left(\boldsymbol{l}\right)+\lambda_{\mathcal{R}}E_{\text{Close}}\left(X\right) \\ & +\lambda_{\text{U}}\underbrace{\sum_{i=1}^{N_X}\left\|\boldsymbol{x}_i-{\boldsymbol{x}_i}^0\right\|^2}_{E_{\text{User}}(X)} \end{aligned}$$

where λ_η, λ_{P}, $\lambda_{\mathcal{R}}$, and λ_{U} are weights, and E_{User} ensures that the initial alignment provided by the user is retained.

To make the model resilient to missing boundary information over large sections of the surface, boundary candidate selection was made robust. With reference to §3.2.7,

[9] Column vectors are used here for consistency; row vectors were used in [109].

[10] $f_0 = 0.15\text{mm}^{-1}$, $b = 2$, and $T = 4$.



(a) (b) (c) (d) (e)

Figure 4.11: (a) The cranial model surface control mesh (top) and Doo-Sabin subdivision surface (bottom) with annotations for the right hemisphere (red), left hemisphere (green), frontal cortex (yellow), falx cerebri (red and green junction), and base of the brain (gray). (b) The default model surface in an example fetal ultrasound image (axial view, from below). (c) The user initialised model surface (from above). (d), (e) The deformed model surface (axial and oblique views).

null labels $\boldsymbol{\zeta}^i$ were introduced for each l_i, E_{FIT} was replaced with $\breve{E}_{\text{FIT}}$:

$$\breve{E}_{\text{FIT}}(X, U, \boldsymbol{l}) = \sum_{i=1}^{N_U} \begin{cases} \left\| \boldsymbol{\phi}_{l_i} - \chi(\boldsymbol{u}_i, X) \right\|^2 + \lambda_\eta \left\| \boldsymbol{\eta}_{l_i} - \nu(\boldsymbol{u}_i, X) \right\|^2, & l_i \neq \zeta_i \\ \kappa, & l_i = \zeta_i \end{cases}$$

where κ is the robust penalty, and the matrix of boundary candidates was extended to define a "position" for each null label ζ_i (§3.2.7). Specifically, with $\breve{\boldsymbol{\phi}}$ denoting the extended matrix of boundary candidate positions, and assuming the null labels are valid column indices, then $\breve{\boldsymbol{\phi}}_{\zeta_i} \leftarrow \chi(\boldsymbol{u}_i, X)$. (A similar definition applies for the extended matrix of boundary candidate normals.)

Initialising X to X^0 and U to a uniform sampling of the model surface domain ($N_U = 1536$), the robust energy was minimised by interleaving discrete and continuous steps. Fixing X and U, the discrete optimisation over $\boldsymbol{l}$ is a first-order `MRF` (§4.3.1) and the algorithm of §4.3.1 was used to find an approximate minimum labelling. To reduce the solving time of `MIN-SUM` and `QPBO`, the label space for each l_i was

restricted to the indices of boundary candidates within a radius of 20mm. Fixing $\boldsymbol{l}$, local continuous optimisation of X and U was performed using `LM`, using the algorithm of §4.2 to perform model coordinate updates. Importantly, E_{P} terms involving null labels were *not* included in the continuous optimisation — only $\boldsymbol{u}_i$ with $l_i \neq \zeta_i$ were updated. While this omission meant that the continuous optimisation was *not* strictly converging to the exact minimum (cf. optimisation of E_{G} in §3.2.7), the quality of the fitted model surface was sufficient for subsequent analysis (Figures 4.11d and 4.11e).

Example results from [109] with parameters $\lambda_\eta = 8$, $\lambda_{\mathrm{P}} = 3$, $\lambda_{\mathrm{U}} = \frac{1}{4}$, $\lambda_{\mathcal{R}} = 1$, and $\kappa = 600$ are shown in Figure 4.12. The complete solving time per image was approximately 2-3 minutes and was dominated by the discrete optimisation step (repeated twice). The reader is referred to [110] and [111] for additional validation and application of the cranial parameterisation.

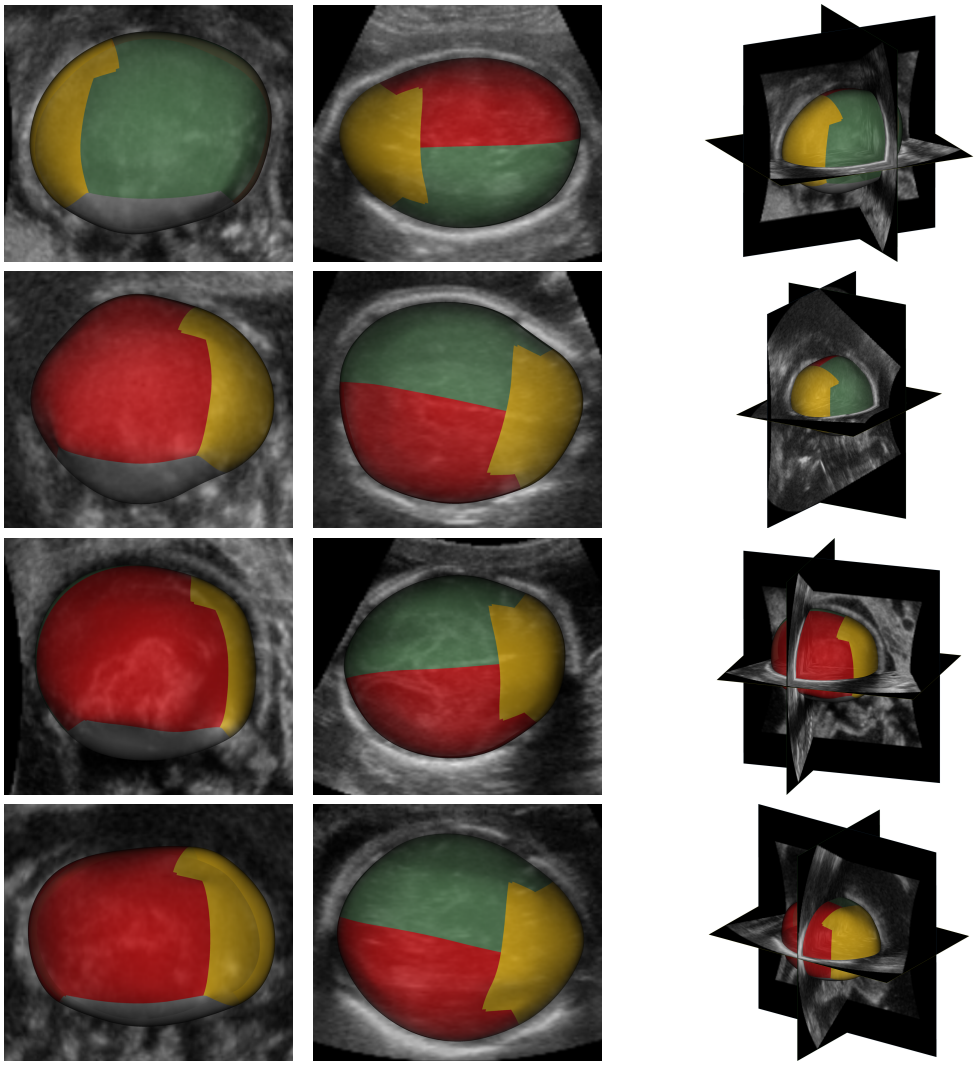
4.4.2 3D Fetal Face Segmentation

The motivation for pursuing 3D model-based fetal face segmentation is to facilitate parametric analysis of facial structures. Subdivision surfaces are particularly useful as model surfaces for this application because they have *local support*, i.e. adjusting the position of a single control vertex results in confined deformation of the model surface. Therefore, by *consistently* fitting a subdivision surface of fixed topology to multiple subjects and images, it has been hypothesised that analysis of the control vertex positions will enable quantitative assessment of differences in facial structure. At the time of writing, quantification of the effects of fetal alcohol syndrome [1] was being investigated by collaborators; this research is ongoing. Therefore, the purpose of this section is to just describe the model surface and surface fitting algorithm.

A Loop subdivision surface with a control mesh of 319 control vertices and 586 faces (493 patches) was used for the model surface (Figure 4.13a). The control mesh was based on a mesh from Turbosquid[11] which was decimated manually in Blender[12].

For each input image, boundary candidates positions $\phi \in \mathbb{R}^{3\times N_\phi}$ were generated by applying a bespoke interactive edge detection algorithm on 2D slices, the details of which are not pertinent to the discussion that follows but are given in [129]. Example slices are shown in Figure 4.13b. To ensure consistent model surface fitting between subjects, a small number of manual correspondences between *fixed* model surface positions $\hat{U}$, corresponding to anatomical landmarks, and a selection of boundary candidates $\hat{\boldsymbol{l}}$, were also specified.

[11] `http://www.turbosquid.com`
[12] `http://www.blender.org`

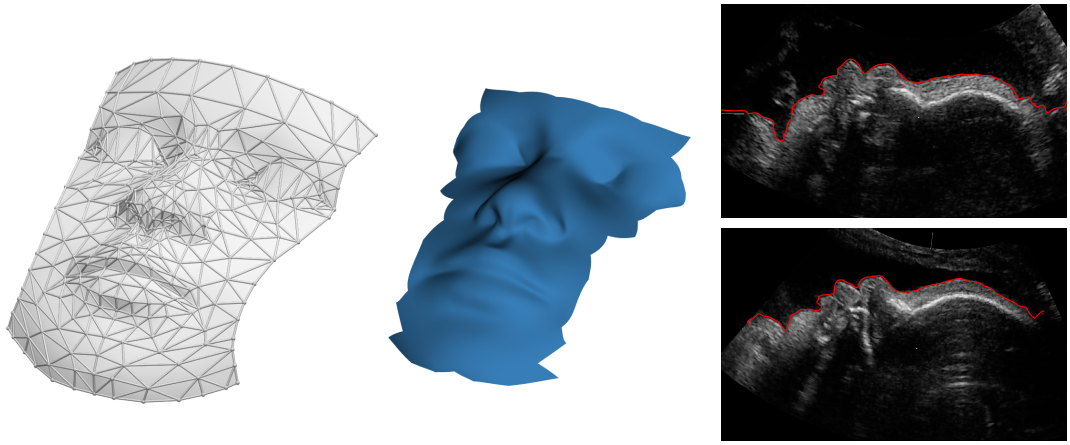


(a) (b) (c)

Figure 4.12: Four example deformed cranial model surfaces (top to bottom) shown from three views [109]. (a) Coronal. (b) Transverse. (c) Oblique.

To start, X was initialised by affine registration of the default model surface using the manual correspondences (Figure 4.14a). Next, $\boldsymbol{l}$ was set to include *all* boundary candidates within a fixed radius r of the model surface (Figure 4.14b), and U was initialised with each $\boldsymbol{u}_i$ set to the model coordinate of the (approximate) closest point on the surface (Figure 4.14c). This is an example of "data-to-model" surface fitting (§4.3.4).

Using thin-plate regularisation (§4.3.2) and robust position errors (§4.3.4), the



(a) (b)

Figure 4.13: (a) Control mesh (gray) and surface (blue) of the model surface. (b) Example 2D slices from 3D fetal ultrasound images with interactively detected edges (red) superimposed.

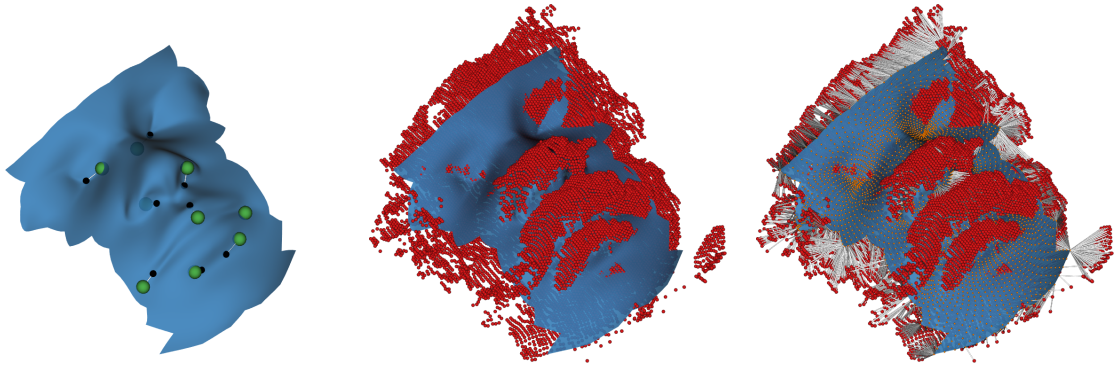
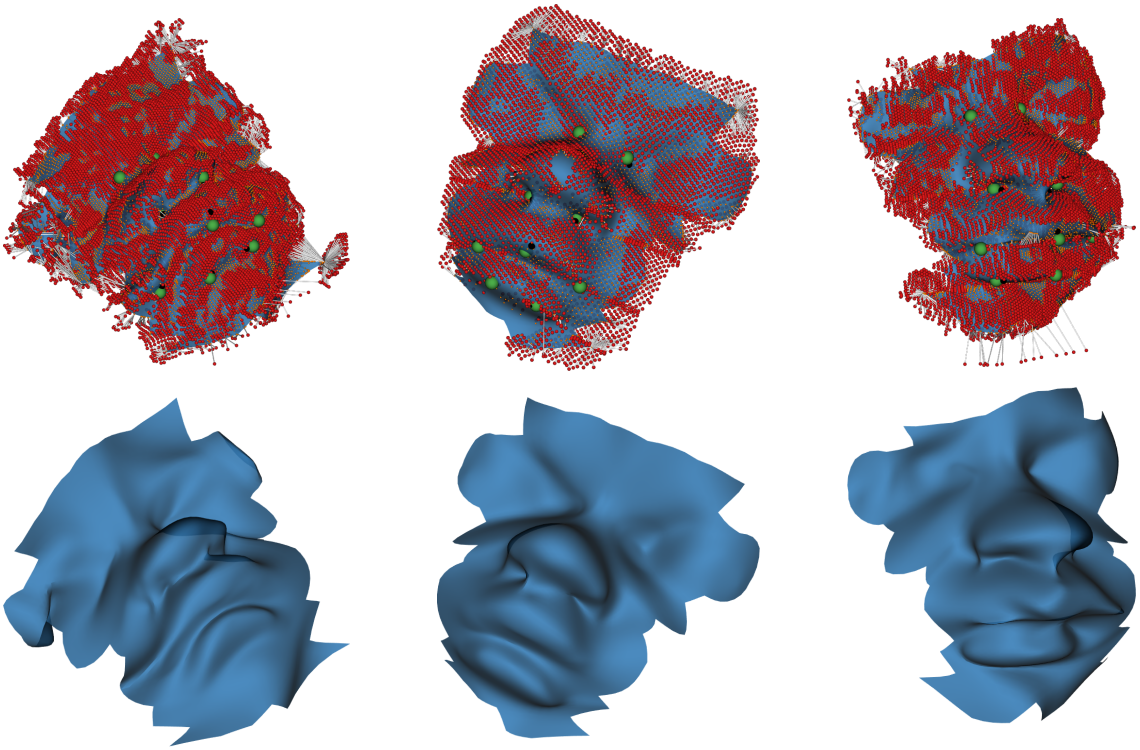


Figure 4.14: Initialisation of X, $\boldsymbol{l}$ and U. (a) The default model surface (blue) was transformed to minimise the distance between manually selected boundary candidates (green) and fixed positions on the model surface (black). (b) *All* boundary candidates (red) within a radius r of the model surface were selected. (c) Model coordinates (orange) were initialised for each boundary candidate.

continuous energy over X and U was defined as:

$$E\left(X, U \mid \boldsymbol{l}\right) = \sum_{i=1}^{N_{\hat{U}}} \rho\left(\lambda_{\hat{\phi}} \left\| \boldsymbol{\phi}_{\hat{l}_i} - \chi\left(\hat{\boldsymbol{u}}_i, X\right) \right\|^2, \delta_{\hat{\phi}}\right) + \sum_{i=1}^{N_U} \rho\left(\left\| \boldsymbol{\phi}_{l_i} - \chi\left(\boldsymbol{u}_i, X\right) \right\|^2, \delta_{\phi}\right) + \lambda_{\mathcal{R}} E_{\mathrm{TP}}\left(X\right)$$

where $N_{\hat{U}}$ is the number of manual correspondences, $\lambda_{\hat{\phi}}$ and $\lambda_{\mathcal{R}}$ are weights, and $\delta_{\hat{\phi}}$ and δ_{ϕ} are the Huber loss function scales. As before, local continuous optimisation was performed using `LM`, using the algorithm of §4.2 to perform model coordinate updates.



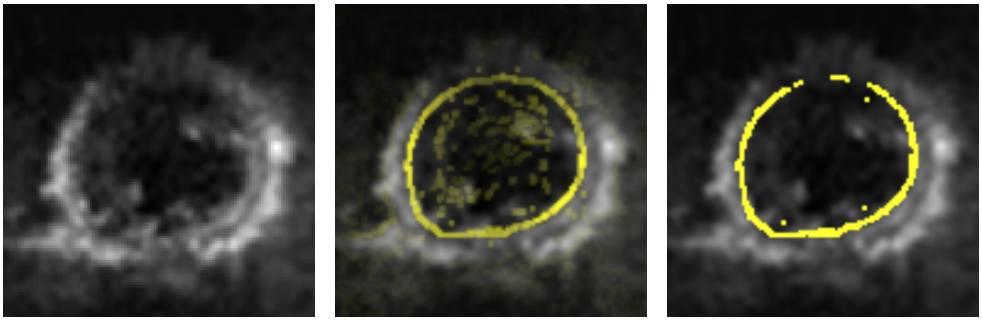
(a) $N_U = 10781$ (b) $N_U = 4957$ (c) $N_U = 9015$

Figure 4.15: Three examples of fitted model surfaces (blue) with all boundary candidates and correspondences shown superimposed (top), and the number of boundary candidates (N_U) given.

Example results with parameters $r = 10\text{mm}$, $\lambda_{\hat{\phi}} = 8$, $\lambda_{\mathcal{R}} = \frac{1}{2}$, $\delta_{\hat{\phi}} = 16$, and $\delta_{\phi} = 2$ are shown in Figure 4.15. Each example was run on an Intel Core i7-4702MQ 2.20GHz processor, using eight threads for residual and Jacobian evaluation. For the three examples, `LM` was run to convergence ($\mu < \mu^0 = 10^{-6}$), requiring 40, 41, and 47 `LM` iterations and taking 44s, 22s, and 45s respectively.

Visual assessment of Figure 4.15 shows that the fitted model surfaces align closely with the selected boundary candidates, especially near the facial midline. The effect of ρ is also demonstrated in Figure 4.15c, where erroneous boundary candidates near the bottom of the face are successfully ignored.

At the time of writing, validation of the algorithm is ongoing. The proposed approach is to fit the model surface to images of normal and abnormal faces, and subsequently use the control vertices as features for a classifier. If the classifier successfully identifies abnormal facial structure, then this will confirm that the model surfaces are (at least approximately) faithful to the underlying anatomical geometry.



(a) (b) (c)

Figure 4.16: (a) Short-axis slice from a 3D echocardiography volume. (b) The probability of a pixel belonging to the endocardium-blood boundary, after non-maximum suppression, superimposed (yellow). (c) Final boundary candidates after thresholding.

4.4.3 3D Echocardiography Left Ventricle Segmentation

The primary purpose of 3D left ventricle (LV) endocardial segmentation is to measure LV volume from echocardiography images. Model-based segmentation is appropriate because calculating the volume of the fitted model surface is trivial, and, as shown in 2D (§3.4.3), it is apt at handling noisy and missing boundary candidates. This section describes the model-based segmentation algorithm that was developed alongside a learning-based edge detection algorithm for LV segmentation [45, 46].

A closed Loop subdivision surface with a control mesh of 24 control vertices and 44 faces (similar to [118]) was used for the model surface (Figure 4.17a).

For a given input image, a Structured Random Forest [44] was used to first regress the probability that each pixel in each short-axis slice belongs to the endocardium-blood boundary. Boundary candidate positions $\phi \in \mathbb{R}^{3\times N_\phi}$ were then generated by thresholding [45] (Figure 4.16).

Let X, U, and $\boldsymbol{l}$ be defined as in §4.3.1. With thin-plate regularisation (§4.3.2) and robust position errors (§4.3.4), the energy over X, U, and $\boldsymbol{l}$ was defined as:

$$E\left(X, U, \boldsymbol{l}\right) = \sum_{i=1}^{N_U} \rho\left(\lambda_\phi \left\| \boldsymbol{\phi}_{l_i} - \chi\left(\boldsymbol{u}_i, X\right)\right\|^2, \delta_\phi\right) + E_{\mathrm{TP}}\left(X\right) \tag{4.10}$$

where λ_ϕ controls the weight of each position error, δ_ϕ is the Huber loss function scale, and continuous optimisation is performed using `LM` ($\mu^0 = 10^{-6}$) with the model coordinate update algorithm of §4.2. (The advantage of using λ_ϕ instead of $\lambda_\mathcal{R}$ is that normalising the weight of the position errors by the number of boundary candidates is slightly simpler to specify.)

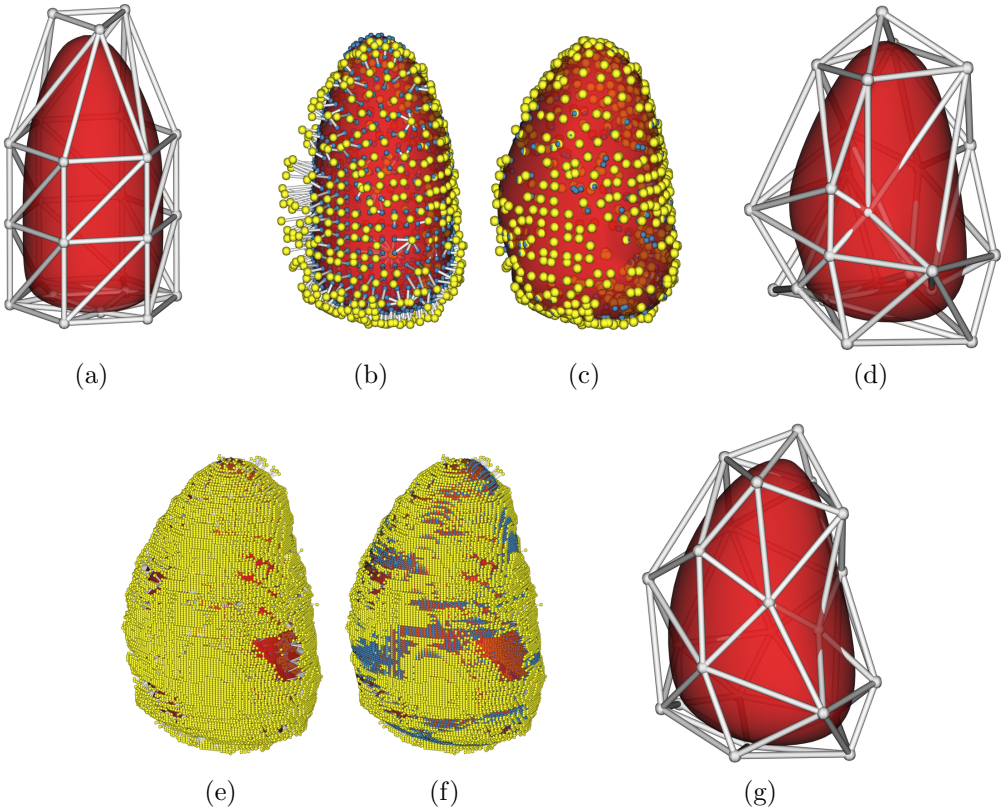


Figure 4.17: The LV model surface and fitting algorithm. (a) The LV model surface (red) was defined by a control mesh (gray) with 24 control vertices. (b)–(d) Model coordinates (blue) were initialised uniformly, boundary candidates (yellow) were selected, and the model surface and coordinates were updated. (e)–(g) Second, *all* boundary candidates within a fixed radius r of the *refined* model surface were selected, model coordinates were reinitialised, and the model surface and coordinates were updated again.

With reference to Figures 4.17b through 4.17g, given an initialisation[13] for X, U was first initialised to a uniform sampling of the model surface domain ($N_U = 794$), and $\boldsymbol{l}$ was set by unconstrained boundary candidate selection (§4.3.1), discarding all $\boldsymbol{u}_i$ with boundary candidates at a distance to the surface exceeding 10mm. Next, with $\lambda_\phi = 2^8$ and $\delta_\phi = 0.025$, (4.10) was minimised with respect to X and U. Following the initial refinement of X, $\boldsymbol{l}$ was then reset to include *all* boundary candidates within a given radius ($r = 5$mm) of the model surface and U was reset with each $\boldsymbol{u}_i$ set to the model coordinate of the (approximate) closest point on the surface (typically, $10^3 < N_U < 10^4$). Finally, X and U were refined again by minimising (4.10) with $\lambda_\phi = 2^{15}/N_U$.

[13]Initialisations were provided by a user using a multi-view graphical user interface.

	d_μ^{ED} (mm)	d_μ^{ES} (mm)	$d_{\mathrm{H}}^{\mathrm{ED}}$ (mm)	$d_{\mathrm{H}}^{\mathrm{ED}}$ (mm)	D_{ED}^*	D_{ES}^*
`TRAINING`	1.12	1.25	6.26	6.66	0.045	0.059
`TESTING-1`	1.99	2.17	9.11	8.70	0.103	0.128

Table 4.2: Summary of distance errors on `TRAINING` and `TESTING-1`

The segmentation accuracy was assessed on the "Challenge on Endocardial Three-dimensional Ultrasound Segmentation" (CETUS) database, comprising of 30 3D+t echocardiography sequences acquired from three different systems: GE Vivid E9 (4V probe), Philips iE33 (X5-1 probe), and Siemens SC2000 (4Z1c probe)[14]. For the "Training Dataset" (`TRAINING`), sequences for 15 subjects (5 healthy, 5 with previous myocardial infarction, and 5 with dilated cardiomyopathy) ground-truth endocardial surface meshes were available for the end-diastole (ED) and end-systole (ES) frames and used to train the edge detection algorithm [46]. For the other 15 subjects, "Testing 1 Dataset" (`TESTING-1`), ground-truth endocardial surface meshes were *not* publicly available; segmentation accuracy was instead assessed by submitting segmentation results directly to the online platform[15].

The distance segmentation errors for the most performant edge detection configuration [46] are shown separately for `TRAINING` and `TESTING-1` in Table 4.2. For each dataset: d_μ^{ED} (d_μ^{ES}) is the mean (across subjects) of the *mean surface distance* for the ED (ES) frames; $d_{\mathrm{H}}^{\mathrm{ED}}$ ($d_{\mathrm{H}}^{\mathrm{ES}}$) is the mean Hausdorff distance, and D_{ED}^* (D_{ES}^*) is the mean *modified* dice similarity index.

The mean surface distance and modified dice similarity index are defined as follows. Let R and S denote discretised *explicit* representations of the ED segmentation and ground-truth surfaces respectively. The mean surface distance is then given by $\frac{1}{2}\left\{\bar{d}\left(R,S\right)+\bar{d}\left(S,R\right)\right\}$, where $\bar{d}\left(R,S\right)$ is the average distance of each point on R to its closest point on S. Similarly, let A and B denote *volumetric* representations of the ED segmentation and ground-truth surfaces respectively. With $A\cap B$ denoting the intersection of A and B, and $|A|$ and $|B|$ denoting the volumes of A and B, $D_{\mathrm{ED}}^* = 1-2\left|A\cap B\right|/\left(|A|+|B|\right)$. ($D_{\mathrm{ES}}^*$ is defined similarly.)

The corresponding volumetric segmentation errors are shown in Table 4.3, where $V_{\mathrm{BIAS}}^{\mathrm{ED}}$ ($V_{\mathrm{BIAS}}^{\mathrm{ES}}$) is the mean (across subjects) volume error for the ED (ES) frames, and V_σ^{ED} (V_σ^{ES}) is the standard deviation.

At the time of writing, for `TESTING-1`, in comparison to nine other algorithms the model-based segmentation framework ranked: 1st for d_μ^{ED}, d_μ^{ES}, D_{ED}^*, and D_{ES}^*; 8th for

[14] `http://www.creatis.insa-lyon.fr/Challenge/CETUS/databases.html`
[15] `http://www.creatis.insa-lyon.fr/Challenge/CETUS/evaluation.html`

	$V_{\text{BIAS}}^{\text{ED}}$ (ml)	V_{σ}^{ED} (ml)	$V_{\text{BIAS}}^{\text{ES}}$ (ml)	V_{σ}^{ES} (ml)
TRAINING	-0.45	7.80	-6.45	8.80
TESTING-1	1.03	21.70	-10.48	9.74

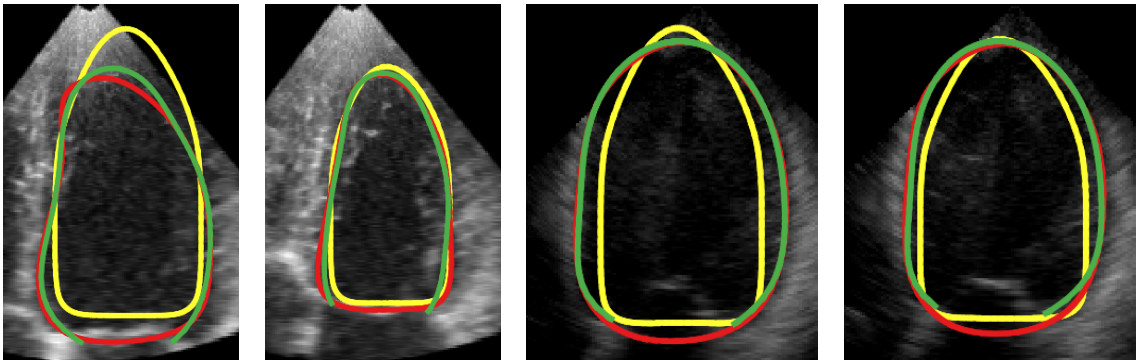
Table 4.3: Summary of volumetric errors on TRAINING and TESTING-1

d_{H}^{ED}; 7$^{\text{th}}$ for d_{H}^{ES}; 2$^{\text{nd}}$ for $V_{\text{BIAS}}^{\text{ED}}$; 7$^{\text{th}}$ for V_{σ}^{ED}; 6$^{\text{th}}$ for $V_{\text{BIAS}}^{\text{ES}}$; 2$^{\text{nd}}$ for V_{σ}^{ES}, and 4$^{\text{th}}$ overall. For context, the lowest V_{σ}^{ED} and V_{σ}^{ES} were 10.69ml and 6.79ml and were achieved by different algorithms. Comparative results for TRAINING were not publicly available. The best overall algorithm — ranked by a combination of segmentation scores and level of automation — was [11], which uses a B-spline Explicit Active Surface [10] for the model surface and the difference in local interior and exterior mean intensities to measure model fit [84].

The favourable ranking for d_{μ}^{ED}, d_{μ}^{ES}, D_{ED}^{*}, and D_{ES}^{*} demonstrates the high delineation accuracy of the model-based segmentation framework. Importantly, the (relatively) poor performance in d_{H}^{ED} and d_{H}^{ES} is easily understood by inspecting the ground-truth protocol and surfaces for TRAINING. With reference to Figure 4.18, the ground-truth surfaces are cut at the basal region of the LV so that the segmentation surface is defined up to, but *not* including, the mitral valve. Therefore, d_{H}^{ED} and d_{H}^{ES} are comparatively high because the model-based framework does *not* perform post-hoc detection of the mitral valve or cutting of the segmentation surface; the mitral valve is nearly always included. This also explains the consistent ES volume overestimation ($V_{\text{BIAS}}^{\text{ES}} < 0$, Table 4.3). Development of a robust procedure for mitral valve detection and final surface refinement was not deemed relevant for validation of the model-based segmentation framework and is left as application-specific future work.

4.5 Conclusions

In this chapter, the details for extending 2D generalised model-based segmentation to 3D were described. Loop and Doo-Sabin subdivision surfaces, which extend uniform quadratic B-spline model contours from 2D to 3D, were introduced as C^1 continuous model surfaces. An algorithm for performing model coordinate updates on these surfaces was also presented, enabling joint optimisation of model surface correspondences *and* model surface geometry. Natural 3D extensions for the boundary candidate selection models and surface regularisers introduced in the previous chapter were also described. Finally, the power and flexibility of the 3D model-based segmentation framework, and its components, were demonstrated on three applications: fetal skull



(a) (b) (c) (d)

Figure 4.18: Example long-axis slices from the end-diastole and end-systole frames for the two subjects with the highest end-diastolic volume and end-systolic volume errors. Contours of the initialisation (yellow), final model surface (red), and ground-truth (green) are shown superimposed.

segmentation, fetal face segmentation, and left ventricle segmentation. Qualitative and quantitative assessment was provided where applicable and available.

The salient conceptual contributions of this chapter are:

1. The primary requirements for extending 2D model-based segmentation to 3D are: definition of a continuous model surface, and an algorithm for performing model coordinate updates.

2. Loop and Doo-Sabin subdivision surfaces are useful 3D extensions of uniform quadratic B-splines. Although their definitions are more complex, the details and intricacies can be "abstracted away" so that positions, normals, first derivatives, and regularisation energies are straightforward to evaluate.

3. Model coordinate updates can be performed in 3D by traversing the model surface domain, facilitating *joint* optimisation of model surface correspondences and the model surface geometry.

4. Application-specific 3D model-based formulations are straightforward to construct by choosing the model surface, regulariser, and boundary candidate selection model as appropriate. Importantly, the same continuous and discrete optimisation algorithms introduced in 2D are directly applicable in 3D.

Chapter 5

Multiple Sequence 3D Model-Based Segmentation

In this chapter, a framework to perform model-based segmentation of *multiple* 3D+t sequences while *jointly* optimising an underlying linear basis shape model is presented.

The motivating application is right ventricle (RV) segmentation from 3D+t echocardiography sequences. The structure of the chapter is as follows. First, the reasons for pursuing the framework for RV segmentation are described. Next, the complete model energy and its optimisation are defined and outlined. Application of the framework is then demonstrated for two problems: single subject RV segmentation from multiple 3D+t sequences acquired from different viewpoints, and *joint* multiple subject RV segmentation from single viewpoint 3D+t sequences. The results confirm that the linear basis shape model is an effective model constraint. Furthermore, it is shown that the proposed framework achieves smaller segmentation errors than a state-of-art commercial semi-automatic RV segmentation package.

5.1 Motivation

For 2D+t and 3D model-based segmentation, regularisers such as "as close as possible" (§3.3.2) and thin-plate energy (§4.3.2) have been used to encourage recovery of smooth and physically plausible model contours and surfaces. The regularisers also adequately constrain the model over small regions of missing boundary candidates (e.g. left ventricle segmentation, §4.4.3). However, for large boundary gaps — such as those encountered when acquiring 3D+t ultrasound sequences of the RV — the simple interpolating action of these regularisers is insufficient to achieve an accurate segmentation. Instead, *linear shape models* can be used which define the control

vertices (points) of the model surface (contour) as a weighted sum of *basis shapes* [18, p. 69].

In medical image analysis, the simple definition of an explicit model surface or contour as a fixed number of discrete points has been popular. For this definition, linear basis shape models have been constructed using Principal Component Analysis (PCA) [16, p. 561] on a set of training surfaces which have been aligned semi-automatically [22, 36]. The advantage of these Active Shape Models (ASMs) is that the dimensionality of the model surface (contour) is reduced. The disadvantage is that the resulting parameterisation can be *too* restrictive and prevent modelling of local deformation unseen in the training examples.

To remedy this, *hierarchical* ASMs have been proposed which recover scale- and location-specific linear basis shape models. In [41], this is achieved in 2D by first computing wavelet coefficients of the x- and y- components of the training model contours. The coefficients are then partitioned into *bands* based on scale and spatial location, and PCA is applied to each. In 3D, spherical wavelets with adaptively selected bands [107, 108], Catmull-Clark subdivision wavelets with fixed bands [90], and diffusion wavelets [51] with orthomax PCA [77, 146], have been proposed. Each of these algorithms produces a linear basis shape model which enables "legal" deformations of the dense model points to be specified with a small number of parameters, where each parameter (by design) controls *local* shape deformation only.

A key challenge when constructing any of the aforementioned shape models is acquiring accurate training surfaces which are in *dense* correspondence. Automatically detecting correspondences based on shape features (e.g. positions of high curvature) has been proposed [24, 154], but this is difficult for the RV because of the absence of dense consistent landmarks across the surface, complete boundaries, and increased shape variability [30, 122]. Furthermore, while training surfaces are rigidly aligned using Procrustes analysis [22], local incorrect correspondences can remain which introduce artificial shape variation into the model. Therefore, it is necessary to model the parameterisation differences between the dense points of the training surfaces. In [42], this is achieved by mapping each surface to the unit sphere and optimising (predefined) parameter transformations for each training surface to construct the ASM of *minimum description length* [133].

In the following sections, a framework is described which performs joint 3D segmentation of the RV from multiple 3D+t echocardiography sequences while *simultaneously* optimising an underlying linear basis shape model. This framework is an extension of [28] — where 3D linear basis shape models of animals are learned from 2D exterior

silhouettes — to 3D *segmentation* of multiple subject and multiple view collections of 3D+t echocardiography sequences. The key differences are:

- In [28], the exterior silhouettes of animals are recovered by semi-automatic segmentation such that *all* boundary candidates are valid and a "data-to-model" approach is possible. Here, boundary candidates are derived by simple edge detection necessitating a "model-to-data" approach instead. Furthermore, boundary candidate positions are noisy and missing boundaries are common.

- In [28], independent rigid transformations and shape parameters are modelled for each frame. Here, scales are introduced for each subject and rigid transformations are introduced for each *sequence*. Shape similarity is also enforced between frames which are of the same subject at the same point in time.

Conceptually, the proposed framework is also similar to [158], where in 2D, explicit model contours are simultaneously fitted to multiple images in a sequence and constrained to be of similar shape. In [158], shape similarity is achieved by minimising the nuclear norm of the matrix composed of the x- and y- components of all model contours. Here, shape similar is enforced explicitly using a linear basis shape model.

As will be shown, the described framework is suitable for the proposed application for three reasons. First, a Loop subdivision surface is used for the model surface, which by construction, has a small number of parameters but is flexible and can realise local shape deformations. Second, joint optimisation of all continuous parameters — including the linear basis shape model control vertices and boundary candidate correspondences — mitigates the requirement for accurate training surfaces that are in dense correspondence. Third, it naturally handles missing boundaries.

5.2 Framework Definition

An overview of the complete framework is shown in Figure 5.1. The definitions for the individual framework components, and the specification and optimisation of the model energy, follow.

5.2.1 Input

The input to the framework is a collection of 3D+t sequences from one or more subjects. The 3D echocardiogram of the k^{th} frame of the j^{th} sequence for the i^{th} subject is identified by the triplet (i, j, k) (or ijk) and denoted as $\mathcal{I}^{ijk}$. Assume that

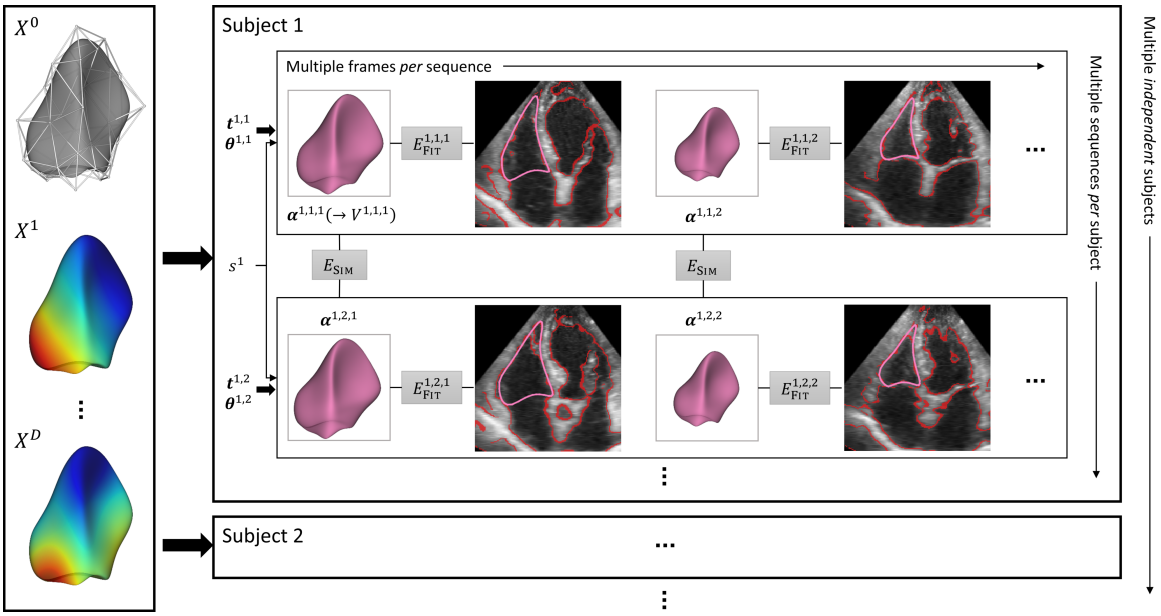


Figure 5.1: The multiple sequence 3D model-based segmentation framework optimises a 3D linear shape model of the RV while performing *joint* segmentation of multiple (temporally aligned) echocardiography sequences acquired from multiple subjects. Each subject i has an associated global scale s^i, and each sequence j has a rigid transformation specified by translation $\boldsymbol{t}^{i,j}$ and rotation $\boldsymbol{\theta}^{i,j}$. For each frame k, the control vertices $V^{i,j,k}$ are defined by a sum of $D+1$ basis shapes $\left\{X^d\right\}_{d=0}^{D}$ with weights $\boldsymbol{\alpha}^{i,j,k}$. E_{SIM} enforces shape similarity between frames from different sequences. (The priors over $\left\{X^d\right\}_{d=0}^{D}$ and $\boldsymbol{\alpha}^{i,j,k}$ are omitted for clarity.)

N^{ijk} boundary candidate positions $\phi^{ijk} \in \mathbb{R}^{3\times N^{ijk}}$ and orientations $\eta^{ijk} \in \mathbb{R}^{3\times N^{ijk}}$ are available in each frame.

5.2.2 Surface Model

A closed Loop subdivision surface with a control mesh of 40 vertices (constructed in Blender[1]) is used for the RV model surface (Figure 5.2). For each image ijk, it is defined by the matrix of control vertices $V^{ijk} \in \mathbb{R}^{3\times 40}$.

To share shape information between sequences and subjects, each V^{ijk} is defined in terms of $D+1$ *unknown* (zero-indexed) basis shapes $\left\{X^d\right\}_{d=0}^{D}$ ($X^d \in \mathbb{R}^{3\times 40}$) and a vector of *unknown* shape coefficients $\boldsymbol{\alpha}^{ijk} \in \mathbb{R}^D$:

$$V^{ijk} \triangleq X^0 + \sum_{d=1}^{D} \alpha_d^{ijk} X^d$$

[1] `http://www.blender.org`

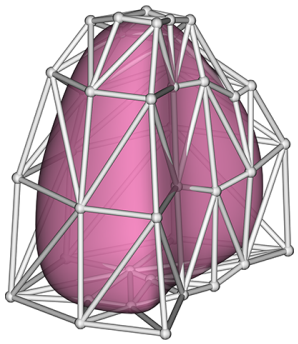


Figure 5.2: The control mesh (gray) defines the RV surface (pink).

5.2.3 Surface Fit

Let $\boldsymbol{l}^{ijk}$ denote the vector of boundary candidate indices and U^{ijk} denote the matrix of model surface correspondences for frame ijk. Introducing s^i as the scale for subject i, and $\boldsymbol{t}^{ij} \in \mathbb{R}^3$ and $\boldsymbol{\theta}^{ij} \in \mathbb{R}^3$ (angle-axis) as the translation and rotation for sequence j respectively, the surface fit is defined as:

$$
\begin{aligned}
&E_{\text{FIT}}^{ijk}\left(U^{ijk}, \boldsymbol{l}^{ijk}, \boldsymbol{\alpha}^{ijk}, s^i, \boldsymbol{t}^{ij}, \boldsymbol{\theta}^{ij}, \left\{X^d\right\}\right) = \\
&\quad \sum_{n=1}^{N^{ijk}} \left\{ \rho\left(\lambda_\phi \left\| \boldsymbol{\phi}_{l_n^{ijk}}^{ijk} - \left(s^i \Theta\left(\boldsymbol{\theta}^{ij}\right) \chi\left(\boldsymbol{u}_n^{ijk}, V^{ijk}\right) + \boldsymbol{t}^{ij}\right) \right\|^2, \delta_\phi \right) + \right. \\
&\qquad \left. \rho\left(\lambda_\eta \left\| \boldsymbol{\eta}_{l_n^{ijk}}^{ijk} - \Theta\left(\boldsymbol{\theta}^{ij}\right) \nu\left(\boldsymbol{u}_n^{ijk}, V^{ijk}\right) \right\|^2, \delta_\eta \right) \right\}
\end{aligned}
$$

where Θ constructs a rotation matrix from $\boldsymbol{\theta}^{ij}$, λ_ϕ and λ_η are weights, and δ_ϕ and δ_η are Huber loss scales (§4.3.4). This is the *robust* oriented surface fit from §4.3.1, *except* that positions on the surface are rigidly transformed using s^i, $\boldsymbol{\theta}^{ij}$ and $\boldsymbol{t}^{ij}$ before computing the position error. Similarly, normals are rotated by $\boldsymbol{\theta}^{ij}$ before computing the orientation error.

5.2.4 Shape Similarity

Consider a subject i. For all sequences j and j' which are temporally aligned with respect to the cardiac cycle (denoted by $\mathcal{A}^i$), shape similarity is encouraged for all common frames (denoted by $\mathcal{K}_{jj'}^i$). The shape similarity energy across *all* subjects is then defined as:

$$
E_{\text{SIM}}\left(\left\{\boldsymbol{\alpha}^{ijk}\right\}\right) = \lambda_{\text{SIM}} \sum_i \sum_{(j,j') \in \mathcal{A}^i} \sum_{k \in \mathcal{K}_{jj'}^i} \left\| \boldsymbol{\alpha}^{ijk} - \boldsymbol{\alpha}^{ij'k} \right\|^2
$$

where λ_{SIM} is a weight.

5.2.5 Surface Regulariser

Thin-plate regularisation (§4.3.2) is applied to each basis shape to encourage recovery of smooth and physically plausible surfaces. However, applying (4.9) directly to each basis has no effect because it is *not* scale-invariant and each subject scale s^i is free. That is, the thin-plate energy can be decreased without affecting the surface fit by arbitrarily shrinking each of the basis shapes X^d and increasing each of the scales s^i accordingly. To prevent this, the thin-plate integral is weighted by the square of the mean of the subject scales $\bar{s}$:

$$E_{\text{PHYS}}\left(X^d, \left\{s^i\right\}\right) = \lambda_{\text{PHYS}} \bar{s}^2 E_{\text{TP}}\left(X^d\right)$$

where λ_{PHYS} controls the regularisation.

Since the shape coefficients also act as scale parameters, they must be treated similarly:

$$E_{\text{REG}}\left(\boldsymbol{\alpha}\right) = \beta \left\|\boldsymbol{\alpha}\right\|^2$$

where β fixes the overall scale of the basis shapes [28].

Finally, to discourage surface self-intersection and folding, the "as close as possible" regulariser is used to maintain tension between adjacent control vertices (4.8):

$$E_{\text{CLOSE}}\left(X^d\right) = \lambda_{\text{CLOSE}} \sum_{(i,j)\in\mathcal{T}} \left\|\boldsymbol{x}_i^d - \boldsymbol{x}_j^d\right\|^2$$

In practice, this term is only active at initial stages of fitting; the weight λ_{CLOSE} is gradually reduced to zero.

5.2.6 Complete Energy

Given $\left(\left\{\phi^{ijk}\right\}, \left\{\eta^{ijk}\right\}\right)$, the complete energy over the basis shapes $\left\{X^d\right\}$, model coordinates $\left\{U^{ijk}\right\}$, boundary candidate indices $\left\{\boldsymbol{l}^{ijk}\right\}$, shape coefficients $\left\{\boldsymbol{\alpha}^{ijk}\right\}$, and rigid transformation parameters $\left(\left\{s^i\right\}, \left\{\boldsymbol{\theta}^{ij}\right\}, \left\{\boldsymbol{t}^{ij}\right\}\right)$ is:

$$\begin{aligned} E = &\sum_{ijk} E_{\text{FIT}}^{ijk}\left(U^{ijk}, \boldsymbol{l}^{ijk}, \boldsymbol{\alpha}^{ijk}, s^i, \boldsymbol{t}^{ij}, \boldsymbol{\theta}^{ij}, \left\{X^d\right\}\right) + E_{\text{SIM}}\left(\left\{\boldsymbol{\alpha}^{ijk}\right\}\right) \\ &+ \sum_{d=0}^{D} \left\{E_{\text{PHYS}}\left(X^d, \left\{s^i\right\}\right) + E_{\text{CLOSE}}\left(X^d\right)\right\} + \sum_{ijk} E_{\text{REG}}\left(\boldsymbol{\alpha}^{ijk}\right) \end{aligned} \tag{5.1}$$

5.2.7 Optimisation Schedule

Given $\left\{\boldsymbol{l}^{ijk}\right\}$, local continuous optimisation of (5.1) is performed using `LM` with the model coordinate update algorithm of §4.2.

Perpendicular boundary candidate selection is used to set $\boldsymbol{l}^{ijk}$ (§4.3.1). Specifically, V^{ijk} is sampled according to U^{ijk}, and the positions and normals are transformed using s^i, $\boldsymbol{\theta}^{ij}$, and $\boldsymbol{t}^{ij}$ before applying (4.6). To prevent surface folding, a boundary candidate is excluded from consideration if the line joining it to the surface point intersects the surface. Furthermore, the selected boundary candidate and its correspondences are discarded if the weighted position and orientation cost exceeds a threshold γ.

A template geometry $\hat{V}$ and affine initialisations $\left\{V^{ijk}\right\}$ are required to initialise the basis shapes $\left\{X^d\right\}$, shape coefficients $\left\{\boldsymbol{\alpha}^{ijk}\right\}$, rigid transformation parameters $(\{s^i\}, \{\boldsymbol{\theta}^{ij}\}, \{\boldsymbol{t}^{ij}\})$ and boundary candidate indices $\left\{\boldsymbol{l}^{ijk}\right\}$.

To start, $D = 2$ with $X^0 = \hat{V}$, $X^1 = [0]$, and $\boldsymbol{\alpha}^{ijk} = [1]$. The rigid transformation parameters — $\{s^i\}$, $\{\boldsymbol{\theta}^{ij}\}$, and $\{\boldsymbol{t}^{ij}\}$ — are initialised by rigidly aligning X^0 to the first affine initialisation of each sequence. Next, U^{ijk} is initialised to a uniform sampling of the model surface domain ($N_U = 610$), and $\boldsymbol{l}^{ijk}$ is set as described above. (The affine initialisations are discarded after this.) Continuous optimisation and boundary candidate selection are then performed in alternation and repeated as required. U^{ijk} is reinitialised at the start of each iteration of boundary candidate selection.

For higher dimensionality shape models, D is incremented before the continuous optimisation step at each round, setting $X^D = [0]$ and resetting $\boldsymbol{\alpha}^{ijk} = \mathbf{1}$. In the experiments that follow, the final value for D was set based on visual inspection of the segmentation surfaces — more sophisticated model selection algorithms based on reconstruction error or model complexity [42] could be utilised but were not investigated here.

5.3 Experiments

The purpose of this section is to demonstrate the application of the multiple sequence segmentation framework for different use cases, to assess its overall segmentation performance, and to determine the usefulness of the underlying linear shape model. The datasets used for testing and validation are described in the next subsection and the details of each use case follow.

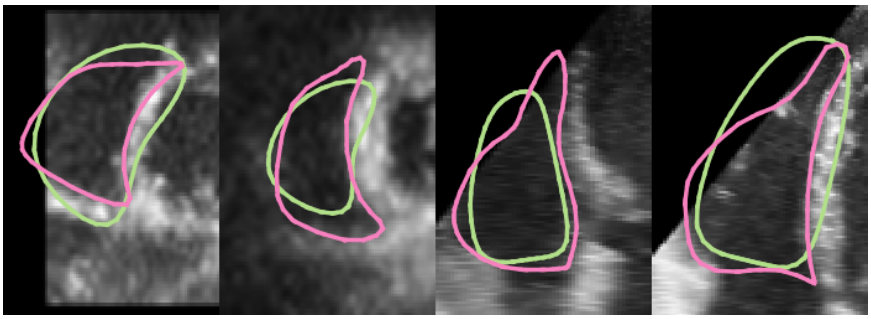


Figure 5.3: Example slices showing initialisations (light green) with the final segmentation surface (pink) superimposed. (Only single slices from each 3D volume are shown for clarity.)

5.3.1 Materials

A Philips iE33 system was used to acquire four-beat 3D+t sequences from 12 subjects who had previously been referred for echocardiography examination. Each 3D+t sequence consisted of 14–28 frames. For 4 subjects, 3–5 sequences were acquired of apical views taken at slightly different transducer positions to ensure good RV coverage. The views were *not* rigidly aligned when presented to the framework. Only single views (from a single transducer position) were recorded for the other subjects.

Boundary candidates were detected in each frame using an isotropic log-Gabor filter with feature asymmetry[2] and non-maximum suppression (§3.4.2).

For the affine initialisations $\{V^{ijk}\}$, a GUI was used which allowed 3D manipulation and scaling of the template in the image domain. Given a new subject and sequence, the user first scaled and translated the template into position in the first frame. For subsequent frames, the initialisation from the previous frames was used so that only minor adjustments by the user were required if necessary. For subsequent sequences, the user initialised the first frame as before, enabling a rigid alignment between the new sequence and previous sequence to be found. The remaining frames of the sequence were then initialised by duplicating and transforming the initialisations from the previous sequence.

Example initialisations are shown in Figure 5.3 (light green contours). As shown, the initialisations do not need to be precise; they only need to be of good enough quality to ensure valid boundary candidates are selected in the first step of optimisation. Semi-automatic initialisation using a "track-to-last" algorithm is therefore possible, although manual initialisations were used here so that the segmentation performance of the framework was assessed in isolation of any other algorithm choices.

[2] $f_0 = 0.06\text{mm}^{-1}$, $b = 2.5$, $T = 4$.

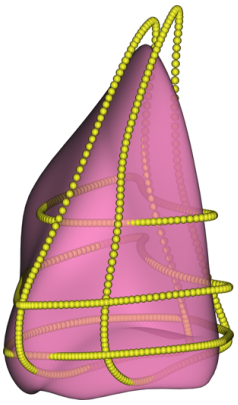


Figure 5.4: For each validation frame, five contours (yellow) were drawn by experts on 2D short- and long-axis image slices. Each contour was discretised into *annotation points* (AP) and the error for each AP was defined as the minimum distance to the segmentation surface (pink).

5.3.2 Validation

For validation, two experts performed manual segmentations of five 2D slices intersecting the RV (2 long-axis, 3 short-axis) for the end-diastole and end-systole frames of each sequence (Figure 5.4). The reasons for this were twofold. First, unambiguous manual delineation of the *complete* RV surface was difficult in most frames because of large missing regions. Second, the primary purpose of the experiments was to assess whether or not the linear basis shape model usefully constrained the segmentation surface without over-constraining it. Contours were sufficient for this.

To measure segmentation error, each manual contour was discretised into *annotation points* (AP) and the minimum distance of each AP to the segmentation surface was first calculated (Figure 5.4). The maximum error (ϵ_{MAX}) for each contour was then defined as the maximum of all AP distances. Similarly, the median error (ϵ_{MEDIAN}) was defined as the median of all AP distances. The error types provide different summaries of segmentation accuracy — ϵ_{MAX} measures the *worst* error whereas ϵ_{MEDIAN} measures the *average*.

5.3.3 Single Subject, Multiple Views

The first use case considered was segmentation of multiple 3D+t sequences acquired from different viewpoints for a single subject ("Single Subject, Multiple Views", `SSMV`). The purpose of this experiment was to examine the segmentation accuracy of the framework and to determine if the subject-specific shape model plausibly interpolates missing boundary information.

#	D	Boundary Candidate Selection			Continuous Optimisation				
		λ_ϕ	λ_η	γ	λ_ϕ	λ_η	λ_{Sim}	λ_{Phys}	λ_{Close}
1	2	1	0.5	2	1	0.0625	8	64	64
2	3	·	·	·	1	·	·	·	32
3	3	·	·	·	4	·	·	·	0

Table 5.1: [SSMV] Alternation parameters.

For this use case, only the 4 subjects with 3–5 sequences were segmented and each was processed separately. The total number of frames segmented jointly for each of the subjects was 44, 75, 88, and 121 respectively. Two-parameter subject-specific shape models ($D = 3$) were deemed sufficient to obtain visually acceptable segmentations. Three rounds of alternation were performed with parameters given in Table 5.1. Other parameter settings were: $\delta_\phi = 1$, $\delta_\eta = 0.125$, and $\beta = 0.5$.

After the final round of joint optimisation, the output surface at each frame was used as an initialisation for independent segmentation *without* the linear basis shape model. The purpose of this was to directly test the efficacy, with respect to segmentation accuracy, of the underlying linear basis shape model for each frame. This approach provides a *baseline* for the segmentation accuracy achievable with only thin-plate regularisation but starting with a good initialisation and *identical* boundary candidates.

5.3.4 Multiple Subjects, Single Views

The second use case considered was the segmentation of multiple 3D+t sequences acquired from single viewpoints for multiple subjects ("Multiple Subjects, Single View", MSSV). The purpose of this experiment was to examine if a linear shape model constraint over multiple subjects could improve segmentation accuracy in comparison to a state-of-art commercial package.

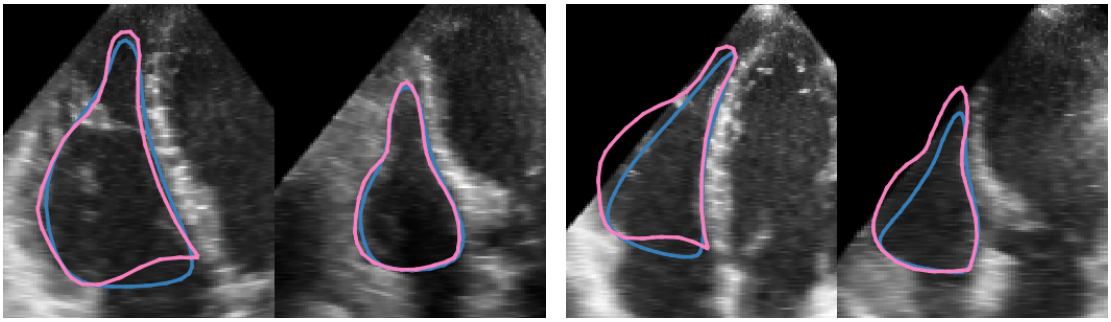
For this use case, the 8 subjects with a single 3D+t sequence were segmented jointly (175 frames in total). A four-parameter shape model ($D = 5$) was deemed sufficient and five rounds of alternation were performed with parameters given in Table 5.2. (Settings identical to SSMV were used for δ_ϕ, δ_η, and β.)

Along with the baseline described in §5.3.3, independent segmentations were also performed by a clinical scientist using TomTec's 4D RV-Function 1.2[3]. This is a semi-automatic method which comprises of 12 steps and requires user input to perform initial contouring and selection of anatomical landmarks.

[3] http://www.tomtec.de/end_users/4d_echo/4d_rv_functionc.html

#	D	Boundary Candidate Selection			Continuous Optimisation				
		λ_ϕ	λ_η	γ	λ_ϕ	λ_η	λ_{SIM}	λ_{PHYS}	λ_{CLOSE}
1	2	1	0.5	2	1	0.0625	0	64	64
2	3	·	·	·	1	·	·	·	32
3	4	·	·	·	1	·	·	·	16
4	5	·	·	·	4	·	·	·	0

Table 5.2: [MSSV] Alternation parameters.



(a) (b)

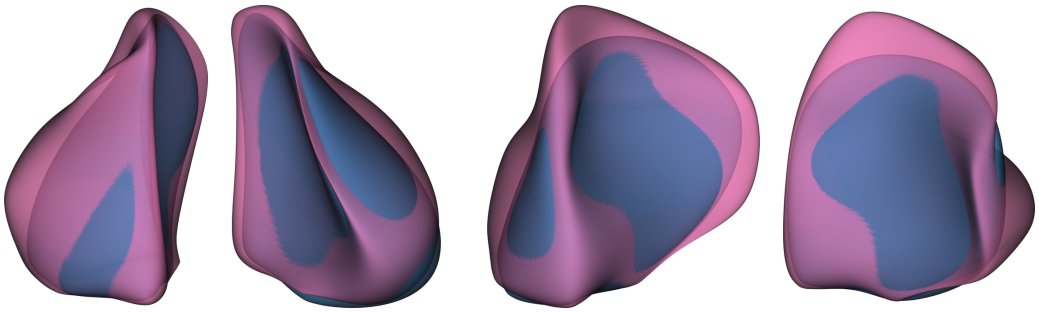
Figure 5.5: [SSMV] Example results for BL (blue) and LBSRV (pink) for two pairs of temporally aligned frames acquired from different views.

5.4 Results and Discussions

5.4.1 Single Subject, Multiple Views

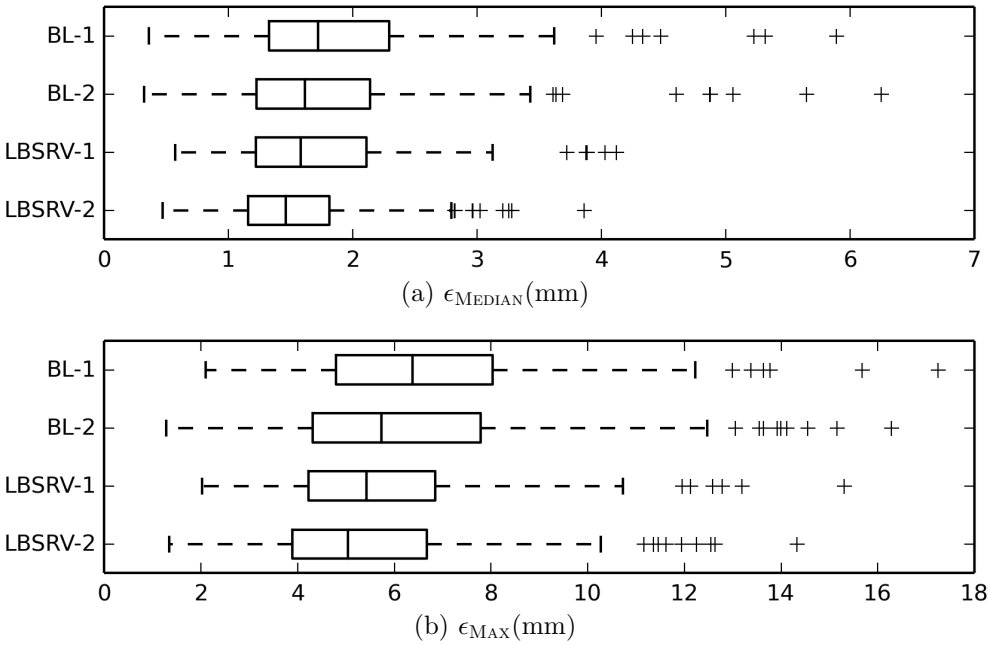
Example slices of echocardiography frames segmented using the framework (Linear Basis Shape RV, LBSRV) and the thin-plate regularisation baseline (BL) for SSMV are shown in Figure 5.5. When all boundaries are available (Figure 5.5a) the segmentation surfaces for both methods reasonably delineate the RV. However, for a temporally aligned frame from a different view (Figure 5.5b), LBSRV implicitly utilises the information from the other views which plausibly interpolates the missing boundary. Conversely, and as expected, BL collapses over the missing region and under-segments the RV. The full segmentation surfaces are shown in Figure 5.6.

Figure 5.7 shows the distribution of both error types (ϵ_{MEDIAN}, ϵ_{MAX}) for all contours, all frames, and against both sets of manual delineations for SSMV. Testing against both sets of contours, LBSRV achieves median ϵ_{MEDIAN} of 1.6mm and 1.4mm respectively (Figure 5.7a). These results are (slightly) lower than the median ϵ_{MEDIAN} achieved by BL — 1.7mm and 1.6mm respectively. For ϵ_{MAX}, LBSRV achieves medians of 5.4mm and 5.0mm which is lower than the 6.4mm and 5.7mm achieved by BL (Figure 5.7b). Note that these results are expected to be very similar because BL was initialised



(a) (b)

Figure 5.6: [SSMV] Two 3D views for each of the segmentation surfaces shown in Figure 5.5b with BL (blue) and LBSRV (pink) shown superimposed.



(a) ϵ_{MEDIAN}(mm)

(b) ϵ_{MAX}(mm)

Figure 5.7: [SSMV] Boxplots of segmentation errors for BL and LBSRV for both sets of manual contours. The y-axis labels are formatted as SegmentationMethod-ManualContourSetIndex.

from LBSRV. However, the combination of these results do support the hypothesis that a low-dimensional linear shape model usefully constrains the segmentation surface *without* over-constraining it.

In Figure 5.7b it can be seen that for some frames the maximum error was greater than 10mm. To understand this, slices from the two worst frames are shown in Figure 5.9. Both frames are of the same subject but are from sequences acquired from different views. From the manual delineations it is clear why ϵ_{MAX} is high — LBSRV under-segments the RV when the mitral valve is open because there are no

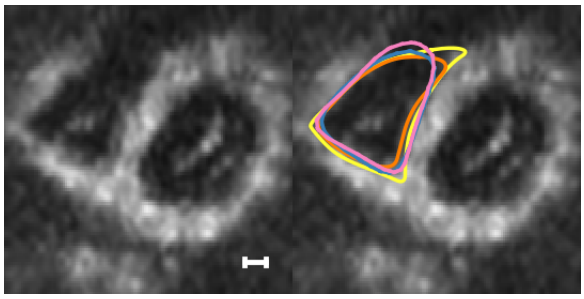


Figure 5.8: [SSMV] (Left) Short-axis slice from the frame with third highest ϵ_{MAX} for LBSRV. (Right) Manual delineations (yellow and orange), BL (blue) and LBSRV (pink) superimposed. Ambiguities arising from missing boundaries show the necessity of multiple manual contours. Both LBSRV and BL agree with one manual contour (orange), but a high ϵ_{MAX} results from the mismatch to the other (yellow). Scale bar (white) is 10mm.

edges to "pull out" the segmentation surface. Instead, it is inexpensive for the surface to assume the concave configuration that is consistent with the mitral valve being closed. Interestingly, for the subsequent frames in each sequence it can be seen that the ventricle boundary might be better defined closer to the apex than the contours in the first frame would suggest. Ambiguities arising from missing boundaries and open regions are also shown in Figure 5.8.

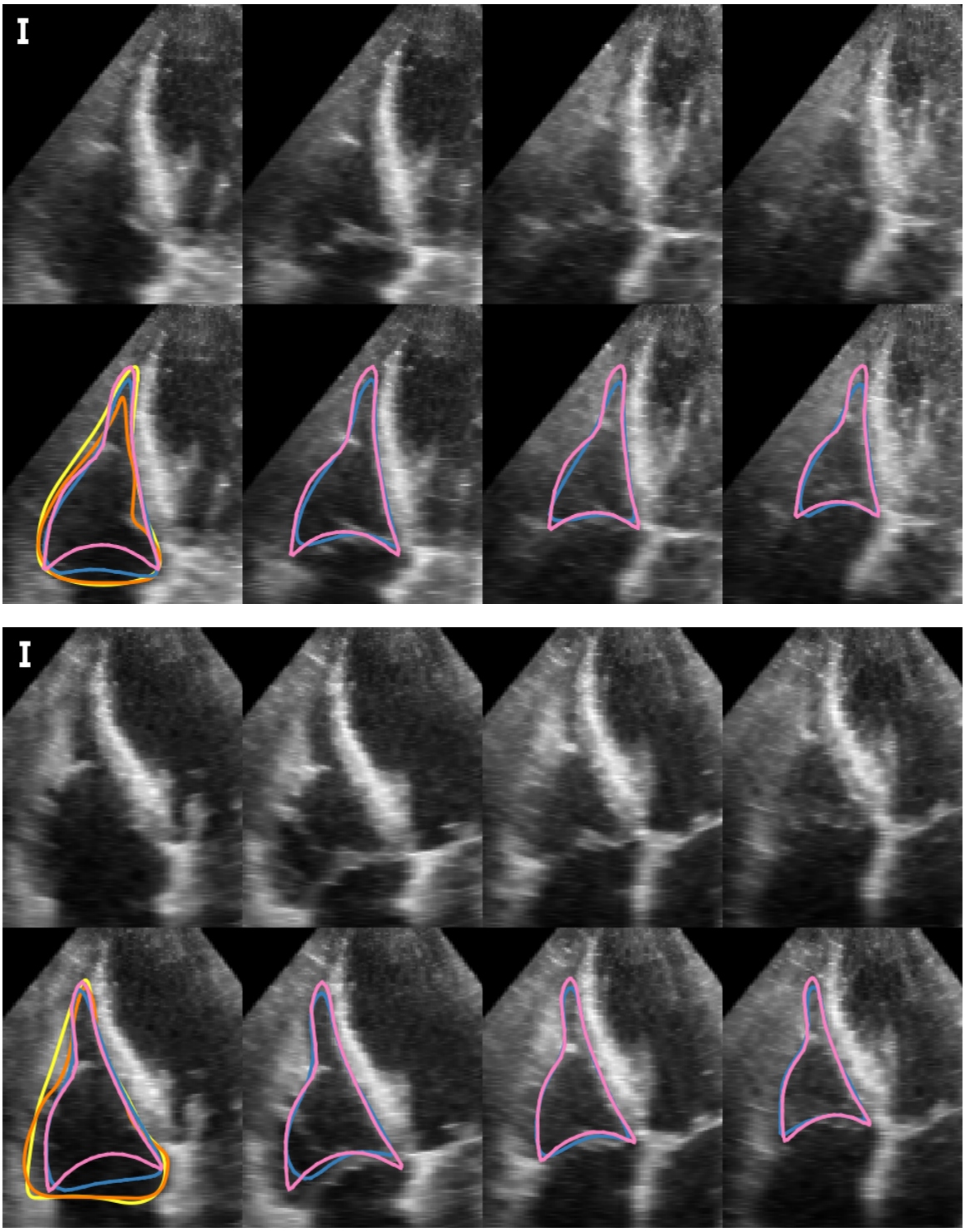
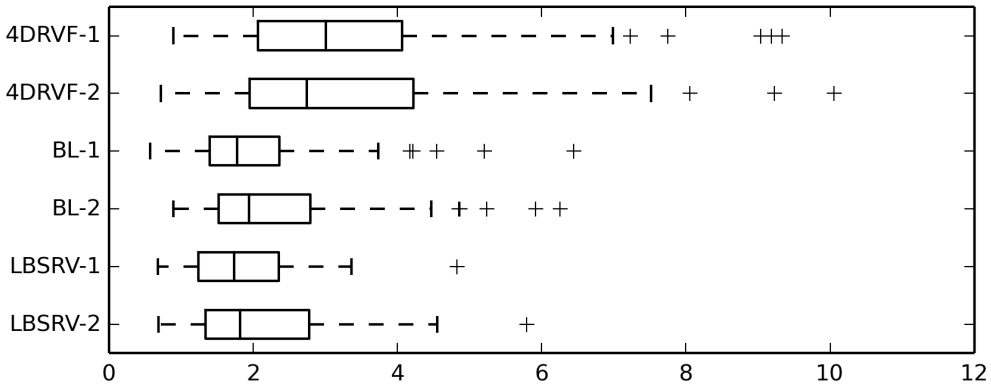


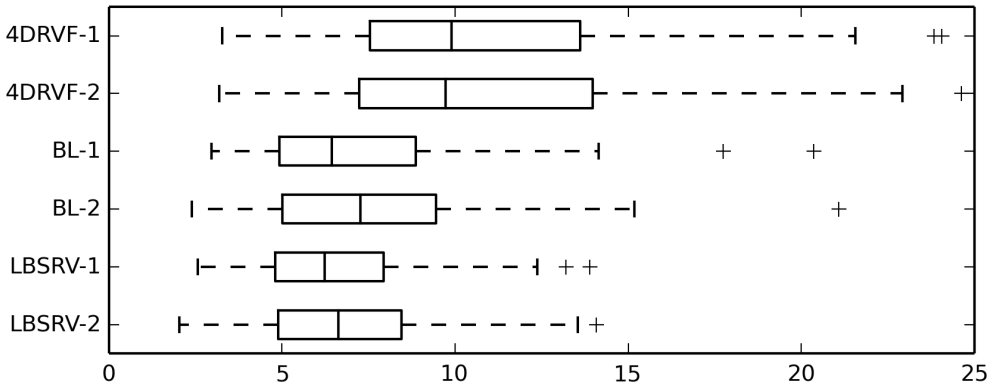
Figure 5.9: [SSMV] (Left) Slices from the two frames with the highest ϵ_{MAX} for LBSRV shown without (top) and with (bottom) delineations. Manual delineations (yellow and orange), BL (blue) and LBSRV (pink) are all shown where available. (Centre-left to right) Slices from subsequent frames in the 3D+t sequence. Scale bar (white) is 10mm.

5.4.2 Multiple Subjects, Single View

Figure 5.10 shows the distribution of errors for all contours, all frames, and against both sets of manual delineations for `MSSV`. Against both sets of contours, `LBSRV` achieves median ϵ_{MEDIAN} of 1.7mm and 1.8mm respectively (Figure 5.10a). These results are lower than `BL` (1.8mm and 2.0mm) and TomTec's 4D RV-Function `4DRVF` (3.0mm and 2.8mm). With reference to Figure 5.10b, `LBSRV` achieves median ϵ_{MAX} values of 6.2mm and 6.6mm which are lower than the 6.4mm and 7.3mm achieved by `BL` and 9.9mm and 9.7mm achieved by `4DRVF`.



(a) ϵ_{MEDIAN}(mm)



(b) ϵ_{MAX}(mm)

Figure 5.10: [`MSSV`] Boxplots of segmentation errors for `4DRVF`, `BL` and `LBSRV` for both sets of contours. The y-axis labels are formatted as `SegmentationMethod-ManualContourSetIndex`.

Comparing timings, `4DRVF` is an interactive method and the average time taken to specify initial contours and anatomical landmarks was 180s per sequence. For comparison, initialisation of each sequence for `LBSRV` took approximately 100s and the final round of continuous optimisation over *all* sequences ($\sim 160 \times 10^3$ parameters, $\sim 500 \times 10^3$ residuals) took under 60 minutes for 50 iterations of LM — an average

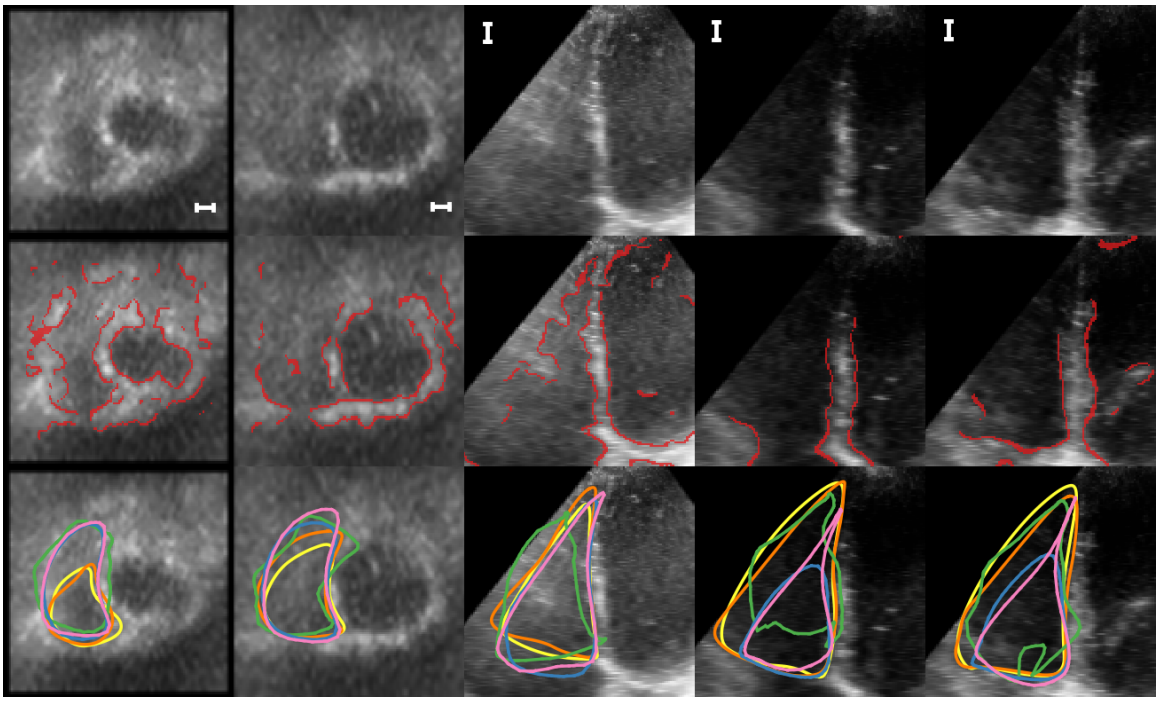


Figure 5.11: [MSSV] Worst slices for LBSRV in terms of ϵ_{MAX}. (Top) Short- and long-axis slices. (Centre) Edges (red) superimposed. (Bottom) Manual delineations (yellow and orange), BL (blue), LBSRV (pink) and 4DRVF (green) superimposed. Scale bar (white) is 10mm.

of 7.5 minutes per subject. Importantly, while 4DRVF is an interactive method, the preset restrictions on user interaction do not allow complete respecification of the segmentation. It was found that spending additional time changing contours and landmarks from their initial positions did not improve segmentation accuracy.

Example delineations for the worst frames in terms of ϵ_{MAX} for LBSRV are shown in Figure 5.11. The first three slices (left to right) are from one subject; the last two are from another.

For the two short-axis slices the segmentation "leakage" arises because of the absence of edges near the delineated RV boundary. While a more sophisticated edge detector or appearance model might detect edges near the manual contour, the actual accuracy of these contours is difficult to guarantee due to the low visibility of the boundary.

For the centre long-axis slice, the high segmentation error arises due to the interpolation of LBSRV over the open mitral valve. Comparing the original slice to the edge map, it can be seen that the faint boundary which guides the manual annotation has not been detected and does not constrain LBSRV.

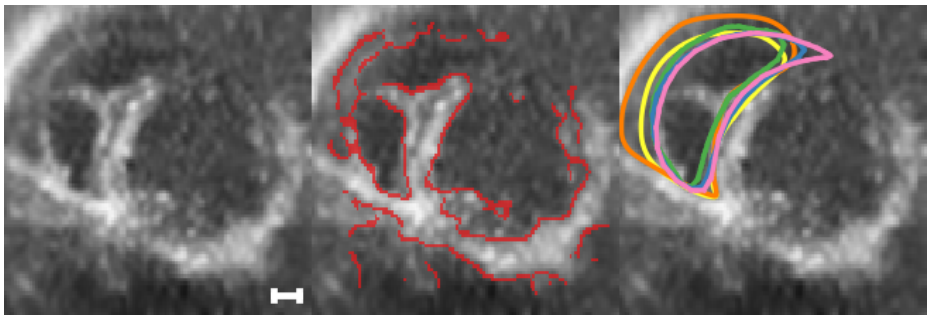
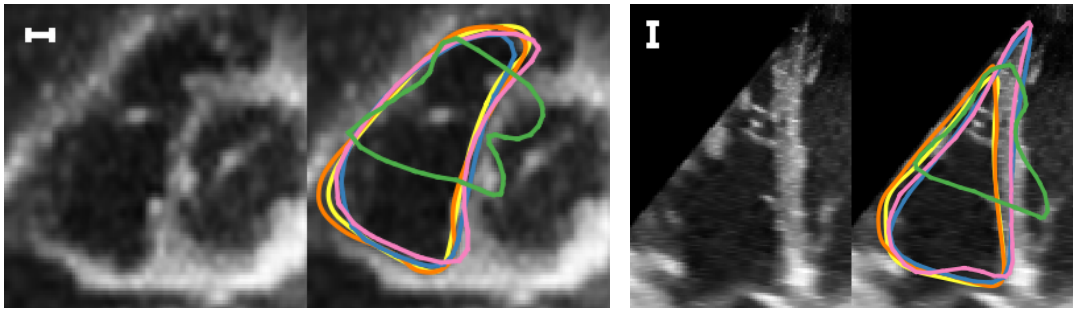


Figure 5.12: [MSSV] Worst slice for LBSRV in terms of ϵ_{MEDIAN}. (Left) Short-axis slice. (Centre) Edges (red) superimposed. (Bottom) Manual delineations (yellow and orange), BL (blue), LBSRV (pink) and 4DRVF (green) superimposed. Scale bar (white) is 10mm.



(a) (b)

Figure 5.13: [MSSV] Two worst slices in terms of ϵ_{MAX} for 4DRVF (green). Manual delineations (yellow and orange), BL (blue), and LBSRV (pink) are provided for comparison. Scale bar (white) is 10mm.

For the final two long-axis slices it can be seen that edges on the interventricular septum (IVS) near the apex were not detected, resulting in under-segmentation. The shape model adopted a collapsed form because this occurred in all frames. This error could have been reduced by adjusting the edge detection parameters. However, for consistency and demonstrative purposes it was decided to retain identical parameters for all frames. Unsurprisingly, these two long-axis slices were also the worst for BL.

The slice from the worst frame (ϵ_{MEDIAN}) for LBSRV is shown in Figure 5.12. The deviation arose because the surface deformed to the papillary muscles instead of the endocardium. As before, this could have been rectified by using an improved edge detector or more careful initialisation.

The two slices from the worst frame (ϵ_{MAX}) for 4DRVF are shown in Figure 5.13. Both Figure 5.13a and Figure 5.13b are examples of total segmentation failures. Examples of comparative quality can also be seen in Figure 5.11.

Finally, Figure 5.13b also provides an example of how stitching artefacts further

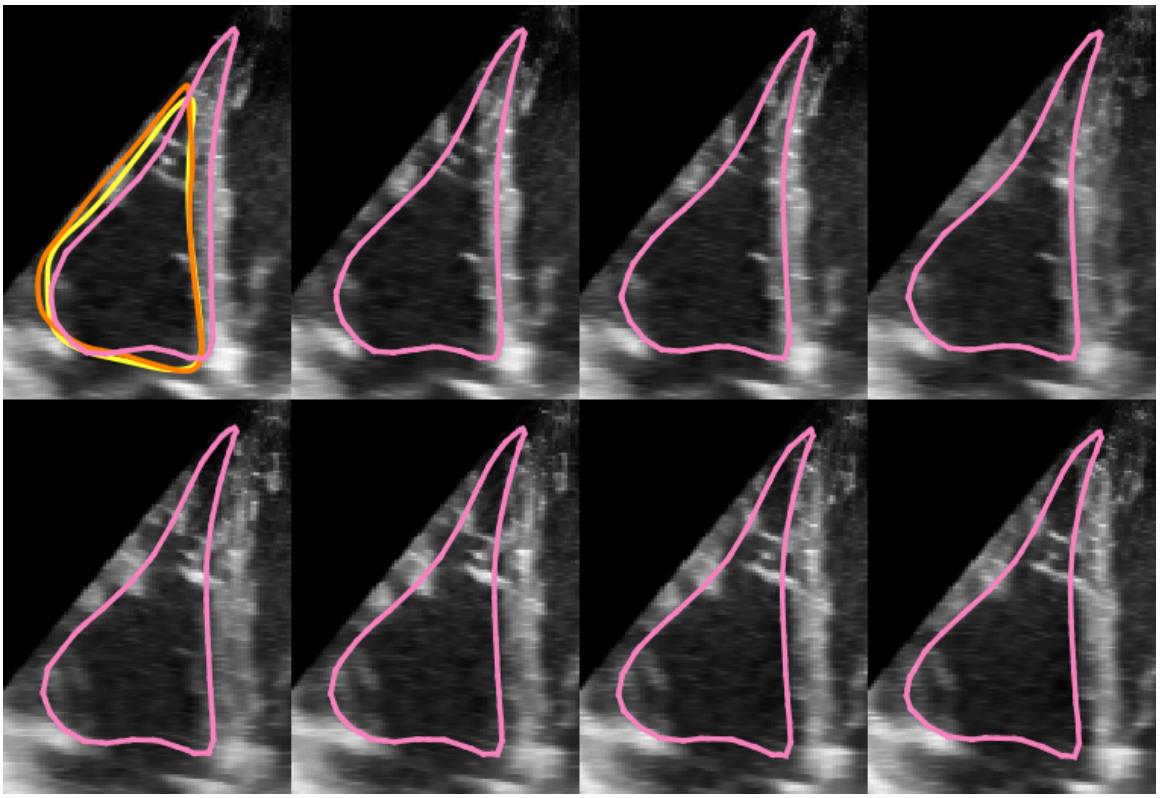


Figure 5.14: [MSSV] (Left to right, top to bottom) Stitching artefacts can be seen by the movement of the IVS in successive adjacent long-axis slices. In the first long-axis slices, LBSRV appears to have over-segmented the RV and leaked past the IVS. Subsequent slices, cropped in the same image coordinates, show that this is not the case.

complicate segmentation validation. In Figure 5.13b, it appears that LBSRV and BL have deformed incorrectly to the left ventricle endocardium. On inspection of adjacent long-axis slices it can be seen that this is not the case (Figure 5.14).

5.5 Conclusions

In this chapter, a framework to perform model-based segmentation of multiple 3D+t sequences while jointly optimising an underlying linear basis shape model was described.

The framework was motivated by difficulties specific to RV segmentation from 3D+t echocardiography sequences. Specifically, large regions of missing boundary candidates are common when imaging the RV due to the relative position of the RV with respect to the transthoracic ultrasound probe. Simple model surface regularisers are incapable of interpolating the missing regions accurately, necessitating the use of a shape model. But, specifying training model surfaces for the RV which are

anatomically consistent is *itself* difficult because of the absence of dense consistent landmarks across the surface, complete boundaries in the exemplar echocardiography frames, and shape variability. The framework presented in this chapter, inspired by the model of [28], removes this requirement in an elegant way: model surfaces for *all* 3D frames are fitted while *simultaneously* optimising a linear basis shape model.

The framework was demonstrated on two problems: single subject RV segmentation from multiple 3D+t sequences acquired from different viewpoints, and *joint* multiple subject RV segmentation from single viewpoint 3D+t sequences. The framework was shown to handle missing boundaries plausibly, and achieved lower segmentation errors than a leading commercial RV segmentation package. The linear shape model was also shown to improve segmentation accuracy over thin-plate surface regularisation alone.

Chapter 6

Surface-Independent Boundary Candidate Selection

In this chapter a boundary candidate selection algorithm which does *not* require a model surface initialisation or estimate is presented. The method is built around a weak parts-based shape model — the Boundary Fragment Model (`BFM`) [115, 138]— which represents an object by sections of its boundary and comprises of two steps. To start, the position and scale of the anatomical object of interest are determined using *only* the structure of the detected boundary candidates. Next, *every* boundary candidate is individually classified as either background or foreground, based on the proximity to the fitted `BFM`.

The structure of this chapter is as follows. First, the motivation for pursuing the framework, and its related work, are presented. Next, the definition and construction of the `BFM` in 3D is given. The object detection and edge classification algorithms are then described successively. Finally, experiments are presented which demonstrate the operation of the entire framework on the task of LV endocardium and epicardium boundary candidate selection. The primary purpose of these experiments is to elucidate the impact of `BFM` parameter choices on its construction and performance. In particular, it is shown that the framework detects the position and scale of the LV with high accuracy, *and* successfully classifies boundary candidates as background, endocardium, or epicardium.

6.1 Introduction

Boundary candidate selection is necessary when only a subset of available boundary candidates are relevant. "Model-to-data" approaches — where boundary candidate indices $\boldsymbol{l}$ are set *given* model coordinates U *and* an estimate of the model surface —

have been used successfully in many of the applications presented so far (§3.4, §4.4, §5.2.7). However, the delineation accuracy of "model-to-data" approaches is implicitly limited because only a small subset of available *valid* boundary candidates are used. This observation motivated the use of "data-to-model" approaches (§4.3.4), where boundary candidates are selected *independently* and before initialisation of model coordinates U. Specifically, for 3D fetal face segmentation (§4.4.2) and single-frame LV segmentation (§4.4.3), *all* boundary candidates within a fixed radius of the (initial or current) model surface were selected. Unfortunately, while this particular "data-to-model" approach was shown to achieve accurate segmentations, it still requires an accurate model surface initialisation or estimate. Eliminating this requirement motivates the work that follows.

In this chapter a boundary candidate selection algorithm is presented which automatically isolates *all* relevant boundary candidates *without* an estimate of the model surface. This is achieved in two steps which utilise *only* the structure of the boundary candidates: *detection* and *delineation*. The structure is captured by a weak parts-based [53] shape model — the Boundary Fragment Model (`BFM`) — where each *edge part* is a set of boundary candidates which describe a section of the anatomical boundary of interest. The position of each part is defined relative to the model centroid, but this position is not fixed and the flexibility is learned during the construction of the model.

The advantage of the `BFM` is that it is parametrised only by its centroid position and scale, enabling its use for global object detection by exhaustive search. To test a given centroid position and scale, the model is "placed" in the image and the support of the edge parts is used to determine the likelihood of the test position. Following [138] and previous work using the `BFM`, a boosted classifier is used to learn which parts best discriminate between different test positions.

Once detection is complete, the `BFM` is used again to drive a separate boundary candidate selection (delineation) step — this has *not* been done previously. The purpose of this step is to isolate boundary candidates corresponding to different anatomical boundaries from irrelevant or spurious edges. To achieve this, the `BFM` is used to derive structural features for each boundary candidate which are then used in a Random Decision Forest (`RDF`) classification framework.

The work presented in this chapter is inspired by the earlier research of [138] and [115], and extends [143] and [145]. It has also been published in *Medical Image Analysis* [144]. The key contributions are generalising the `BFM` to 3D, extending its application from object detection to boundary candidate selection, and interrogating

its operation towards a common medical image analysis problem — segmenting the left ventricle (LV) from 3D echocardiography frames.

6.1.1 Related Work

The use of edges (boundary candidates) for object detection has a strong history in computer vision and a comprehensive overview is given in [139]. The focus of this section is to instead explain how the approach in this chapter relates to current object detection and classification approaches in medical image analysis.

6.1.2 Discriminative Object Detection

Supervised classification algorithms have been utilised for many detection problems in medical image analysis. The purpose of these classification algorithms is to capture the structure of an object class of interest so that an image can be exhaustively searched. For example, `AdaBoost` [56] has been used for detection of liver tumours [121], Random Decision Forests (`RDF`) have been used for organ localisation in CT volumes [40], and support vector machines have been used for hippocampal segmentation in brain MR images [104]. All of these techniques use appearance features for detection and the structure of the detected object is implicitly learned through the combination of the appearance features. In contrast, the `BFM` explicitly models sections of the object boundary which are used to derive features. The choice of which sections to use reflects the variability of the different sections across the object's boundary — sections which vary less are more reliable structural features.

6.1.3 Discriminative Voxel Classification

Segmentation techniques based on discriminative voxel classification have been popular in medical image analysis. These techniques aim to develop a classifier which can label a voxel based on local and contextual appearance information. Again, `RDF` classifiers have been used for myocardium segmentation in 3D ultrasound images based on positional and appearance features [87]. To enforce local segmentation consistency, *auto-context* [152] and *entanglement features* [103] have been proposed which utilise full or partial classification of neighbouring voxels to inform classification at the voxel of interest.

Voxel classifiers are typically translation-invariant by design and the structure of objects of interest are implicitly learned by the choice of appearance features. `RDF` frameworks have been proposed which are not translation-invariant, but an assumption

about the location of the object of interest is made *a-priori* [87]. Furthermore, smooth boundary delineation is difficult because discrimination between classes must be made over a one-voxel boundary [103, 139]. Here, the *delineation* step mitigates this problem by classifying boundary candidates directly — the smoothness of the boundaries is instead determined by the initial edge detection process. Furthermore, the structure of the object is captured by the combination of the fitted `BFM` and the features used by the classifier. As a consequence, the classifier only learns how relevant boundary candidates spatially relate to sections of the fitted model and does not have to implicitly capture the complete structure of the object.

6.2 The Boundary Fragment Model

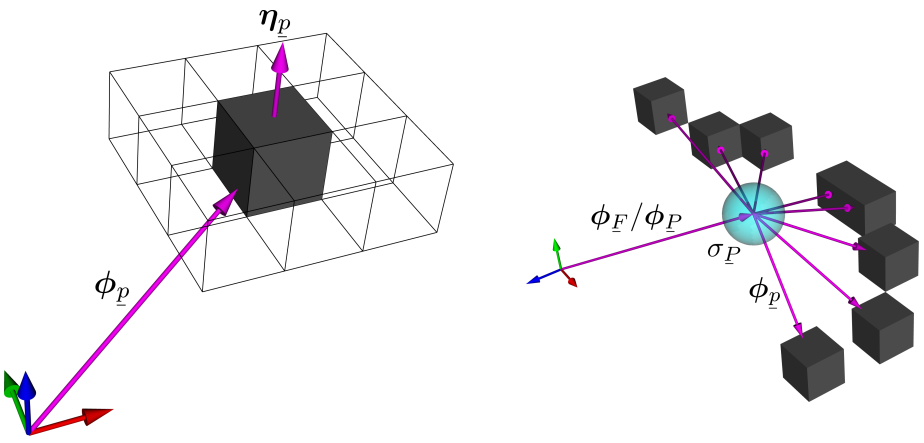
The purpose of the `BFM` is to capture the scale-normalised shape information of an object class using only boundary candidates. The `BFM` presented here draws largely from [138] with the addition that it is generalised to 3D. Therefore, the purpose of this section is to reintroduce the `BFM` with consistent notation, compare the two main previous formulations by Opelt et al. [115] and Shotton et al. [138], and provide the necessary background to understand how the model can be used for detection (§6.3) and boundary candidate selection (§6.4).

6.2.1 Notation

An underlined letter such as $\underline{F}$ or $\underline{p}$ denotes an *object*. An object can have multiple attributes, each of which is denoted by its name *to the left of* the object. For example, the *vector* attribute $\boldsymbol{\phi}$ of $\underline{p}$ is denoted by $\boldsymbol{\phi}_{\underline{p}}$. Sets are unordered unique collections of objects and are denoted by upper case letters with no subscripts e.g. F.

6.2.2 Edge Maps

For a d-dimensional image $\mathcal{I}$ its *edge map* is a binary image that is `true` at positions which correspond to edges in $\mathcal{I}$ and `false` otherwise. Each pixel that contributes to an edge is an *edgel* $\underline{p}$ which has both position and orientation. The edgel's position $\boldsymbol{\phi}_{\underline{p}} \in \mathbb{R}^d$ is the offset of the edge from a known centre (e.g. the image origin) and can be in the coordinates of the quantised image domain or in physical units. The edgel's orientation $\boldsymbol{\eta}_{\underline{p}} \in \mathbb{R}^d$, or $\eta\left(\underline{p}\right)$, is a unit vector perpendicular to the direction of the edge. In 2D, Opelt et al. and Shotton et al. store the orientation as an angle, but a unit vector in the direction of the edge normal is simpler in 3D (Figure 6.1a).



(a) (b)

Figure 6.1: (a) An edgel $\underline{p}$ has a position $\boldsymbol{\phi}_{\underline{p}}$, defined relative to an origin (three axes), and an orientation $\boldsymbol{\eta}_{\underline{p}}$. (b) A fragment $\underline{F}$ (or part $\underline{P}$) has an origin $\boldsymbol{\phi}_{\underline{F}}$ ($\boldsymbol{\phi}_{\underline{P}}$) defined relative to the object centroid (three axes). For each $\underline{p}$ in F (P), $\boldsymbol{\phi}_{\underline{p}}$ is defined relative to the fragment origin. An edge part *additionally* stores a measure of uncertainty for the origin position ($\sigma_{\underline{P}}$, blue sphere).

6.2.3 Edge Fragments

An *edge fragment* $\underline{F}$ is a set F of edgels with a position vector $\boldsymbol{\phi}_{\underline{F}} \in \mathbb{R}^d$. The vector $\boldsymbol{\phi}_{\underline{F}}$ is the offset from the object centroid to the fragment origin. F is the set of edgels — defined relative to the fragment origin — which contribute to a section of the object boundary. An *edge part* $\underline{P}$ is an edge fragment with an additional scalar $\sigma_{\underline{P}}$ that quantifies the uncertainty of $\boldsymbol{\phi}_{\underline{P}}$ (Figure 6.1b). Scale-normalised edge fragments (parts) also contain the scale $s_{\underline{F}}$ ($s_{\underline{P}}$) at which they were acquired.

6.2.4 Distance Between Sets of Edgels

It is necessary to define a notion of distance between sets of edgels in order to construct a `BFM` and use edge fragments for object detection[1].

Given two sets of edgels with the same origin — a template set T and another set E — the chamfer distance of T *to* E at an offset $\boldsymbol{x}$ is defined as:

$$d^{E,\tau}_{\text{CHAM}}(T, \boldsymbol{x}) = \frac{1}{\tau\,|T|} \sum_{\underline{q} \in T} \min\left(\tau, \min_{\underline{p} \in E} \left\| \left(\boldsymbol{\phi}_{\underline{q}} + \boldsymbol{x}\right) - \boldsymbol{\phi}_{\underline{p}} \right\|\right) \tag{6.1}$$

where τ is the upper distance limit and $|T|$ (the cardinality of the set T) is the number of edgels in T. The chamfer distance is the normalised sum of the Euclidean distances

[1] "Distance functions" defined in the remainder of this section are *not* metrics over sets of edgels; the terminology is retained only for convenience and compatibility with [138].

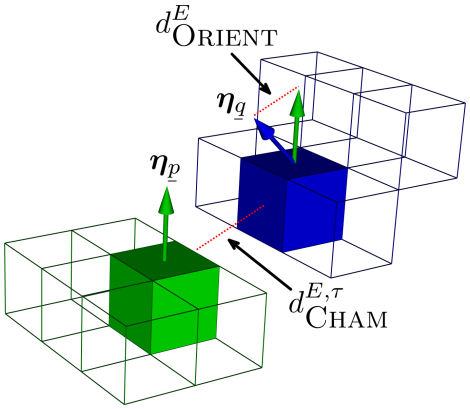


Figure 6.2: Consider two sets of edgels T (blue) and E (green) at some offset $\boldsymbol{x}$. For a given $\underline{q} \in T$, its closest edgel in E is $\underline{p}$. $d^{E,\tau}_{\text{CHAM}}$ measures the average spatial distance from $\underline{q}$ to $\underline{p}$ for *all* $\underline{q} \in T$. Similarly, d^E_{ORIENT} measures the average orientation difference.

from each edgel in T to its closest edgel in E (Figure 6.2). This can be efficiently obtained by precomputing the clipped distance transform $\text{DT}^{E,\tau}(\boldsymbol{x})$ [52]:

$$\text{DT}^{E,\tau}(\boldsymbol{x}) = \min\left(\tau, \min_{\underline{p} \in E} \left\|\boldsymbol{x} - \boldsymbol{\phi}_{\underline{p}}\right\|\right)$$

The chamfer distance only captures spatial distance between edgel sets. To include orientation information Opelt et al. propose dividing the edge maps into eight separate orientation channels and summing the individual chamfer distances from each channel. Shotton et al. propose the *oriented distance* as a continuous extension instead. The oriented distance is the normalised sum of the smallest angular distances from each edgel in T to its closest edgel in E (Figure 6.2):

$$d^E_{\text{ORIENT}}(T, \boldsymbol{x}) = \frac{1}{\pi |T|} \sum_{\underline{q} \in T} \arccos\left(\eta\left(\underline{q}\right) \cdot \eta\left(\text{ADT}^E\left(\boldsymbol{\phi}_{\underline{q}} + \boldsymbol{x}\right)\right)\right) \tag{6.2}$$

where $\text{ADT}^E(\boldsymbol{x})$ is the argument distance transform which returns the edgel in E closest to the position $\boldsymbol{x}$ and is computed at the same time as the distance transform:

$$\text{ADT}^E(\boldsymbol{x}) = \arg\min_{\underline{p} \in E} \left\|\boldsymbol{x} - \boldsymbol{\phi}_{\underline{p}}\right\|$$

The definition of oriented distance in (6.2) differs from the one given by Shotton et al. in two ways. First, a dot product and arccos are required to calculate the smallest angular distance since $\eta\left(\underline{q}\right)$ is a unit vector and not an angle. Second, the complete edgel orientation has been preserved so that the orientation vector describes the direction of positive intensity change. In typical computer vision applications

this is undesirable as it imposes a constraint on the intensity relationship between the object of interest and the background. However, for medical image analysis this orientation information is important for distinguishing between edges that describe inner and outer anatomical boundaries.

Finally, to simultaneously describe the spatial and angular distance of T to E, *oriented chamfer matching* is used which is the weighted sum of the chamfer distance and oriented distance [138]:

$$d^{E,\lambda}_{\mathrm{OCM}}(T, \boldsymbol{x}) = (1-\lambda)\, d^{E,\tau}_{\mathrm{CHAM}}(T, \boldsymbol{x}) + \lambda d^{E}_{\mathrm{ORIENT}}(T, \boldsymbol{x})$$

where λ controls the contribution of the two distance functions.

6.2.5 Model Construction

The `BFM` of an object is a set of scale-normalised edge parts which represent regions of the object using boundary candidates. This section describes the construction of a `BFM` given a training set of edge maps which contain the object class of interest.

6.2.5.1 Fragment Generation

In order to generate the `BFM` it is first necessary to generate candidate edge fragments. Given training data where each object has a corresponding bounding box, Shotton et al. propose randomly selecting sets of edgels within the object bounding box. With the same training data requirement Opelt et al. instead randomly select edgel seeds in the bounding box and grow the edge fragments based on the connectivity of the edge map. An approach similar to Shotton et al. is used since artefacts in ultrasound images can prevent continuity of edges along boundaries.

Given a set of d-dimensional edge maps, assume each object of interest has a corresponding bounding box $\boldsymbol{b} \in \mathbb{R}^{2\times d}$ which holds the minimum and maximum coordinates along each dimension. The centre of the bounding box $\bar{\boldsymbol{b}}$ is taken as the centroid of the object. To generate a candidate edge fragment, a random *fragment box* $\boldsymbol{r} \in \mathbb{R}^{2\times d}$ with centre $\bar{\boldsymbol{r}}$ is generated within $\boldsymbol{b}$. To prevent generating overly sparse or dense fragments, the fragment box is accepted only if the density of edgels is within some bounds $(\xi_{\mathrm{MIN}}, \xi_{\mathrm{MAX}})$. If accepted, this fragment box defines a new edge fragment $\underline{F}$ (Figure 6.3). The set of edgels within $\boldsymbol{r}$ form the edgel set F, where the positions of the edgels are defined relative to $\bar{\boldsymbol{r}}$. The fragment offset vector is given by $\boldsymbol{\phi}_{\underline{F}} = \bar{\boldsymbol{r}} - \bar{\boldsymbol{b}}$. While this process can also generate edge fragments that do not correspond to a boundary from the object of interest, such fragments may still be useful for object detection and delineation and can be part of the `BFM`.

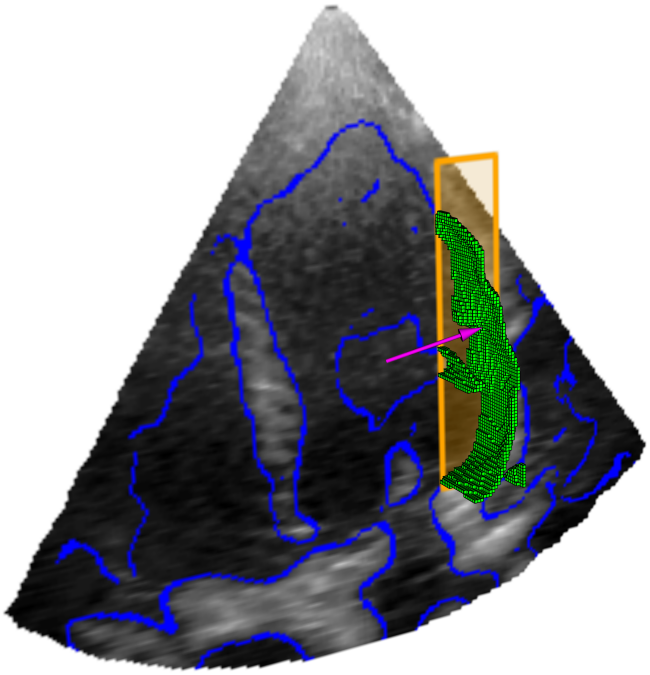


Figure 6.3: The projection of a fragment box (orange) onto an echocardiogram slice with edgels (blue) overlaid. The edgels within the box (green) form the edge fragment $\underline{F}$ ($\boldsymbol{\phi}_{\underline{F}}$ is shown in purple).

To generate a scale-normalised fragment, $\underline{F}$ is assumed to be acquired at a scale s with $\log s \sim \mathcal{U}(-\log s_r, \log s_r)$, where $\mathcal{U}$ is a uniform distribution and s_r is the maximum scale variation to model. The scale is stored as $s_{\underline{F}}$ and the edgel set and fragment offset are normalised by dividing by $s_{\underline{F}}$. Although the assigned scale is randomly selected, the clustering process described in the next section ensures that edge fragments with correct scale estimates form cluster centres.

Additionally, Shotton et al. propose applying small random translations and rotations to F to improve the tolerance of the model to variations in pose but this is omitted here.

6.2.5.2 Fragment Clustering

With a set of N candidate scale-normalised edge fragments generated from the training data, clustering is required to reduce the set to a manageable number and ensure that correctly scale-normalised fragments are retained. The hierarchical clustering approach proposed by Shotton et al. clusters the fragments first by the structure of their edgel sets and second by their position. This is reviewed here for completeness.

To cluster the fragments based on structure, the *symmetric appearance distance* between two scale-normalised fragments $\underline{F}_i$ and $\underline{F}_j$ is defined as:

$$d_{i,j} = d_{\text{OCM}}^{s_i F_i, \lambda}\left(s_i F_j, \mathbf{0}\right) + d_{\text{OCM}}^{s_j F_j, \lambda}\left(s_j F_i, \mathbf{0}\right)$$

where sF denotes a scaled edge fragment, $s_k \equiv s_{\underline{F}_k}$ is the scale of fragment k, and $\mathbf{0}$ denotes a vector of zeros. The symmetric appearance distance between all scale-normalised fragments defines an $N \times N$ distance matrix and appearance clusters are determined using the `K-Medoids` algorithm [79].

The first stage of clustering separates the edge fragments into groups with similar edgel structure based around the fragment origin (Figure 6.4). However, edge fragments with similar structure can describe different sections of the object (Figure 6.5). The (variable number of) positional clusters are found by clustering $\boldsymbol{\phi}_{\underline{F}}$ using `MeanShift` [35].

An edge part $\underline{P}$ is constructed from the fragments in each resulting `MeanShift` cluster $\mathcal{C}$ if the number of fragments is above a given threshold[2]. The medoid edgel set P is determined using the symmetric appearance distance and `K-Medoids` algorithm ($K = 1$). The final part offset $\boldsymbol{\phi}_{\underline{P}}$ is the mean scale-normalised fragment offset from all fragments within the cluster, and $\sigma_{\underline{P}}$ is the standard deviation of the radial distances:

$$\boldsymbol{\phi}_{\underline{P}} = \frac{1}{|\mathcal{C}|}\sum_{\underline{F}\in\mathcal{C}} \boldsymbol{\phi}_{\underline{F}}, \quad \sigma_{\underline{P}} = \sqrt{\frac{1}{|\mathcal{C}|}\sum_{\underline{F}\in\mathcal{C}} \left\|\boldsymbol{\phi}_{\underline{F}} - \boldsymbol{\phi}_{\underline{P}}\right\|^2}$$

The set of all edge parts $\mathcal{M}$ is the `BFM` for an object.

6.3 Object Detection

This section describes how the `BFM` is used to automatically locate the centroid and scale of an object in a candidate edge map. The detection framework presented here is similar to Shotton et al., except that the calculation of a "ground-truth" scale is made explicit. Furthermore, understanding the detection framework is necessary to understand boundary candidate selection in §6.4.

6.3.1 Individual Part Responses

For a candidate object centroid $\boldsymbol{x}$ and scale s, the *part response* is the minimum distance of part $\underline{P}$ to an edgel set E within a given search region. Given the set of

[2] A conservative threshold of 2 was used in the experiments that follow.

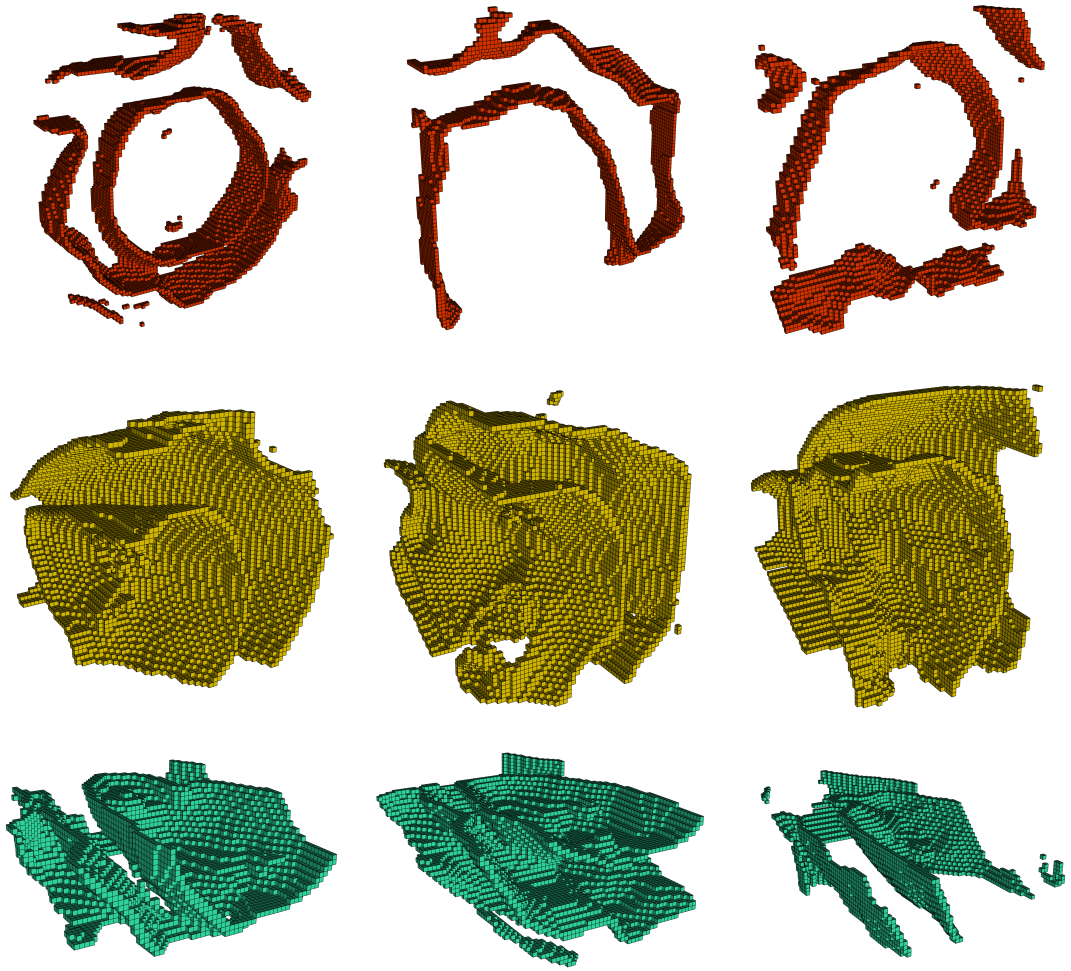


Figure 6.4: Three example appearance clusters (along each row) of fragments derived from the LV.

offsets $\mathcal{R}_P = \left\{\boldsymbol{x} \colon \boldsymbol{x} \in \mathbb{R}^d, \|\boldsymbol{x}\| \leq k\sigma_{\underline{P}}\right\}$, where k controls the relative size of the search region, the part response is[3]:

$$v^{\lambda}\left(\underline{P}, \boldsymbol{x}, s\right) = \min_{\boldsymbol{x}_P \in \mathcal{R}_P} d^{E,\lambda}_{\mathrm{OCM}}\left(sP, \underbrace{\boldsymbol{x} + s\boldsymbol{\phi}_{\underline{P}} + \boldsymbol{x}_P}_{\text{Part origin}}\right) \tag{6.3}$$

(The part response can also include a penalty based on the squared length of $\boldsymbol{x}$ to prevent the part moving too far from its initial position [138].)

[3]In practice the search region is quantised over the discrete domain of the image.

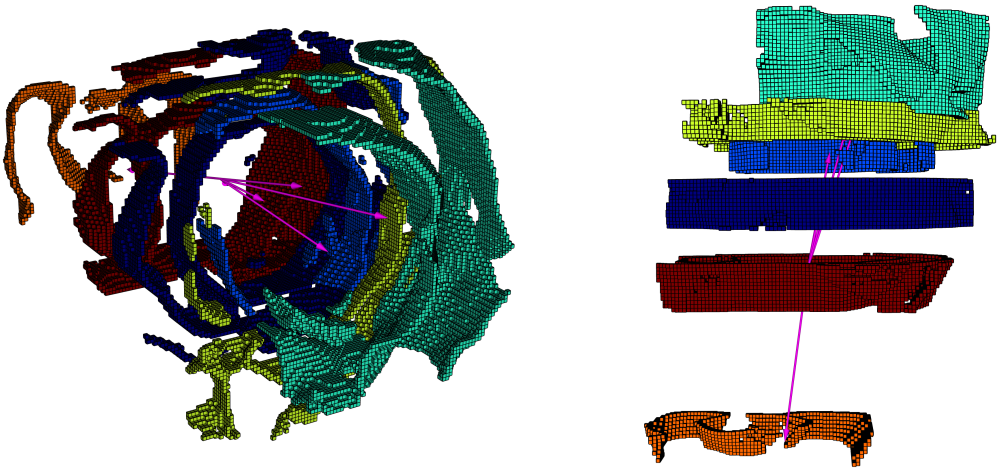


Figure 6.5: Short-axis fragments (shown at a fixed scale) can have similar structure but describe different sections of the endocardium and epicardium (two views).

6.3.2 The Boosted Detection Classifier

Given the `BFM` $\mathcal{M}$, a boosted classifier (`GentleBoost` [56]) can be constructed which confirms or rejects a candidate centroid $\boldsymbol{x}$ and scale s based on multiple part responses (Figure 6.6):

$$H(\boldsymbol{x}, s) = \operatorname{sgn}\left(\sum_{i=1}^{M} h_{\theta_i,a_i,b_i}\left(v^{\lambda}\left(P_{\underline{\omega}_i}, \boldsymbol{x}, s\right)\right)\right) = \operatorname{sgn}\left(\hat{H}(\boldsymbol{x}, s)\right) \tag{6.4}$$

where M is the number of weak learners, $\boldsymbol{\omega}$ is the vector of M selected (possibly non-unique) part indices, sgn is the sign function, and h_{θ_i,a_i,b_i} is a *decision stump* with threshold θ_i which returns a_i if $v^{\lambda}\left(P_{\underline{\omega}_i}, \boldsymbol{x}, s\right) \geq \theta_i$ and b_i otherwise [56].

Positive and negative examples of centroid and scale positions are required to train the boosted classifier. Given a training set of edge maps with the boundaries delineated, the centroid is taken as the centre of the boundary bounding box $\bar{\boldsymbol{b}}$. The ground-truth scale s_g is then determined from a set of candidate scales S:

$$s_g = \arg\min_{s \in S} \sum_{\underline{P} \in \mathcal{M}} v^{\lambda}(\underline{P}, \boldsymbol{x}, s)$$

where the part responses are taken on the edge map with *only* the boundary edgels. A scaled grid pattern is then used on the full edge map to provide positive and negative centroid examples, ensuring that the classifier is tolerant to small changes in position and scale. Positive positions are taken from the set:

$$\left\{\bar{\boldsymbol{b}} + s_g \delta_1 \boldsymbol{z} \gamma_1^{z_s} : \boldsymbol{z} \in \{-1, 0, +1\}^d, z_s \in \{-1, 0, +1\}\right\}$$

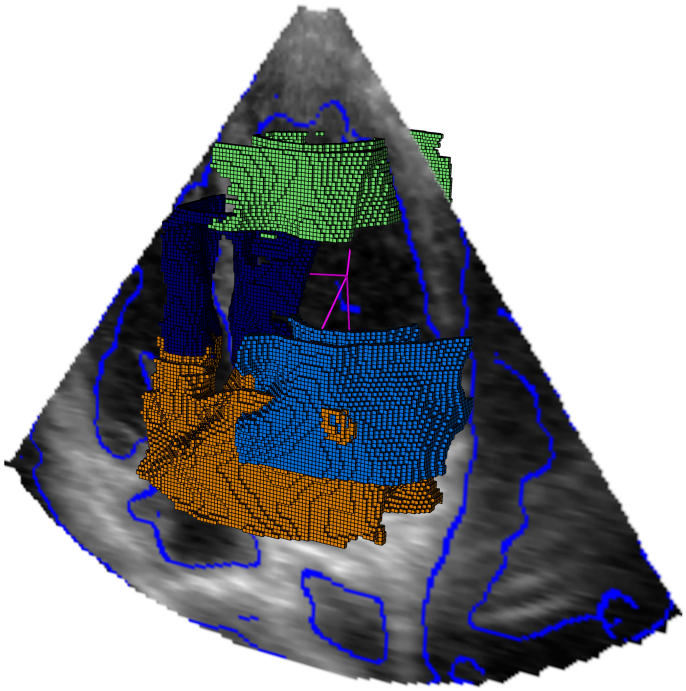


Figure 6.6: Each *part response* is generated by calculating the oriented chamfer distance of each part to the test edge map (blue). Intuitively, each part response measures how well the part fits into the test edge map for the given centroid position and scale.

where δ_1 controls the scale-normalised grid spacing and γ_1 controls the variation in scale. Negative positions are taken from a similar set except with a larger grid spacing δ_2, larger scale variation γ_2, and excluding the centre position. The grid pattern causes $H(\boldsymbol{x}, s)$ to respond positively to positions and scales close to (and at) the desired centroid and scale, and negatively elsewhere.

6.3.3 Sliding Window Detection

The centroid and scale classification score can be interpreted as a probability using a logistic sigmoid transformation:

$$P(\boldsymbol{x}, s) = \left(1 + e^{-\hat{H}(\boldsymbol{x},s)}\right)^{-1} \tag{6.5}$$

Because of the localisation properties of the classifier, it does not need to be evaluated at every possible centroid position and scale. Instead, (6.5) is evaluated at equally spaced positions in space and scale and `MeanShift` is used to find the modes of $P(\boldsymbol{x}, s)$. For single object detection, the mode with the highest probability is taken as the object centroid and scale.

6.4 Delineation / Boundary Candidate Selection

Following object detection, it is still necessary to perform object delineation (boundary candidate selection). Previously, segmentation approaches using the `BFM` have been based on a single principle — edge parts which supported the final centroid position likely correspond to sections of the object boundary. Opelt et al. propose back-projecting the boundary fragments that supported the detected object centroid to form a "sketch" of the object. This sketch is then filled and morphological operations are used to clean-up and produce a final segmentation. An alternative approach, presented previously in [145], determines the relevance of edgels in the candidate edge map by inspecting their spatial and orientation distance to the supporting edge parts. The supporting edge parts are defined as those that provided a positive contribution to the boosted classifier (6.4) and are positively supported by the image ($b_i > 0$).

The method presented in this section extends previous approaches by learning a classifier which can use all edge parts to assign a label to each edgel. Given the detected centroid position and scale, multiple edgel classes can be identified using a single `BFM`. This is useful for separating inner and outer boundary responses, for example.

6.4.1 Edgel Features

Given the detected centroid and scale of an object, it is desirable to assign a label $l \in \mathcal{L}$ to each edgel. Learning a classifier to perform this task requires defining features which are easily evaluated at each edgel. The features presented in this section depend *only* on the edgel's structural relationship to the `BFM` and detected centroid position.

The most obvious feature for an edgel $\underline{q}$ is its spatial distance to an edge part $\underline{P}$:

$$g^{\tau}_{\text{CHAM}}\left(\underline{q}, \underline{P}\right) = \frac{1}{\tau}\min\left(\tau, \min_{\underline{p}\in P} \| \underbrace{\left(\boldsymbol{x} + s\boldsymbol{\phi}_{\underline{P}} + \boldsymbol{x}_P\right)}_{\text{Part origin}} + s\boldsymbol{\phi}_{\underline{p}} - \boldsymbol{\phi}_{\underline{q}} \|\right) \tag{6.6}$$

where $\boldsymbol{x}$ and s are the detected object centroid position and scale respectively, $\boldsymbol{x}_P$ is the part offset determined from the part response in (6.3), and τ controls the sensitivity. (The quantities $\boldsymbol{x}$, s, and $\boldsymbol{x}_P$ are strictly arguments of g^{τ}_{CHAM} but are omitted for clarity.)

The spatial distance can be efficiently evaluated by transforming the position of

edgel $\underline{q}$ into the coordinate system of $\underline{P}$:

$$\begin{aligned} g^{\tau}_{\text{CHAM}}\left(\underline{q},\underline{P}\right) &= \frac{1}{\tau}\min\left(\tau, \min_{\underline{p}\in P}\left\|s\boldsymbol{\phi}_{\underline{p}} - \left(\boldsymbol{\phi}_{\underline{q}} - \boldsymbol{x} - s\boldsymbol{\phi}_{\underline{P}} - \boldsymbol{x}_P\right)\right\|\right) \\ &= \frac{1}{\tau}\min\left(\tau, \frac{s}{s_{\underline{P}}}\min_{\underline{p}\in P}\left\|s_{\underline{P}}\boldsymbol{\phi}_{\underline{p}} - \frac{s_{\underline{P}}}{s}\left(\boldsymbol{\phi}_{\underline{q}} - \boldsymbol{x} - s\boldsymbol{\phi}_{\underline{P}} - \boldsymbol{x}_P\right)\right\|\right) \\ &= \frac{1}{\tau}\frac{s}{s_{\underline{P}}}\min\left(\frac{s_{\underline{P}}}{s}\tau, \min_{\underline{p}\in P}\left\|s_{\underline{P}}\boldsymbol{\phi}_{\underline{p}} - \boldsymbol{\phi}'_{\underline{q}}\right\|\right) \\ &= \frac{1}{\tau}\frac{s}{s_{\underline{P}}}\min\left(\frac{s_{\underline{P}}}{s}\tau, \text{DT}^{s_{\underline{P}}P,\tau_{\text{MAX}}}\left(\boldsymbol{\phi}'_{\underline{q}}\right)\right) \\ &= \min\left(1, \frac{1}{\tau}\frac{s}{s_{\underline{P}}}\text{DT}^{s_{\underline{P}}P,\tau_{\text{MAX}}}\left(\boldsymbol{\phi}'_{\underline{q}}\right)\right) \end{aligned} \tag{6.7}$$

where $s_{\underline{P}}$ is the acquired scale of the edge part $\underline{P}$, $\text{DT}^{s_{\underline{P}}P,\tau_{\text{MAX}}}$ is the clipped distance transform of the part at its acquired scale, $\boldsymbol{\phi}'_{\underline{q}}$ is the transformed position of edgel $\underline{q}$:

$$\boldsymbol{\phi}'_{\underline{q}} \triangleq \frac{s_{\underline{P}}}{s}\left(\boldsymbol{\phi}_{\underline{q}} - \boldsymbol{x} - s\boldsymbol{\phi}_{\underline{P}} - \boldsymbol{x}_P\right) \tag{6.8}$$

and τ_{MAX} is the maximum possible τ given by:

$$\tau_{\text{MAX}} \triangleq \max_{s,\underline{P}\in\mathcal{M}}\left(\frac{s_{\underline{P}}}{s}\tau\right)$$

The definition of $\tau_{\max}$ allows for the efficient calculation of (6.6) using (6.7) because the distance transform of $\underline{P}$ is only computed once.

The distance defined by (6.6) is nearly identical to (6.1) except that (6.6) is measuring the distance of a single edgel from the candidate edge map to a scaled edge part, while (6.1) is measuring the distance of a set of edgels to another set of edgels.

The oriented distance of $\underline{q}$ to $\underline{P}$ is therefore defined analogously to (6.2). Using (6.8):

$$g_{\text{ORIENT}}\left(\underline{q},\underline{P}\right) = \frac{1}{\pi}\arccos\left(\eta\left(\underline{q}\right)\cdot\eta\left(\text{ADT}^{s_{\underline{P}}P}\left(\boldsymbol{\phi}'_{\underline{q}}\right)\right)\right) \tag{6.9}$$

Combining (6.6) and (6.9) gives the oriented chamfer matching distance of $\underline{q}$ to $\underline{P}$:

$$g^{\lambda}_{\text{OCM}}\left(\underline{q},\underline{P}\right) = (1-\lambda)\, g^{\tau}_{\text{CHAM}}\left(\underline{q},\underline{P}\right) + \lambda g_{\text{ORIENT}}\left(\underline{q},\underline{P}\right)$$

For a given edgel $\underline{q}$, multiple features are defined by taking the oriented chamfer matching distance of $\underline{q}$ to *every* part $\underline{P}_i$ in $\mathcal{M}$:

$$f^{\lambda}_i\left(\underline{q}\right) = g^{\lambda}_{\text{OCM}}\left(\underline{q},\underline{P}_i\right) \tag{6.10}$$

The minimum distance of an edgel to any part describes the edgel's overall proximity to the fitted parts:

$$f^{\lambda}_{\text{MIN}}\left(\underline{q}\right) = \min_{\underline{P}\in\mathcal{M}} g^{\lambda}_{\text{OCM}}\left(\underline{q}, \underline{P}\right) \tag{6.11}$$

Both (6.10) and (6.11) can be evaluated at multiple values of λ to vary the emphasis of spatial distance over orientation.

Finally, the orientation of the edgel relative to the detected centroid gives a measure of roundness and is useful for discriminating between edgels which contribute to inner and outer boundaries:

$$f_{\text{ORIENT}}\left(\underline{q}\right) = \frac{1}{\pi}\arccos\left(\eta\left(\underline{q}\right)\cdot\frac{\boldsymbol{\phi}_{\underline{q}} - \boldsymbol{x}}{\left\|\boldsymbol{\phi}_{\underline{q}} - \boldsymbol{x}\right\|}\right) \tag{6.12}$$

For n different λ, $n(|\mathcal{M}|+1)+1$ features — $n|\mathcal{M}|$ from (6.10), n from (6.11), and one from (6.12) — make up the *feature vector* for the edgel.

6.4.2 Multiclass Edgel Classifier

A Random Decision Forest (`RDF`) classifier is used to classify each edgel based on its feature vector. The `RDF` classifier is an ensemble classifier made up of multiple decision trees [16, 58, 139]. A decision tree consists of successive decision stumps which branch left or right depending on the value of an individual feature. To classify an edgel, its feature vector is calculated and passed down each decision tree until it reaches a leaf node. The final class distribution for the edgel is taken as the average of the class distributions (learned from training data) at each leaf node on each decision tree. The final classification is taken as the label with the highest probability.

Labelled edge maps are required to train the edgel classifier. For each labelled edge map the object centroid and scale are first detected and the edgel features are subsequently determined to construct a full feature matrix. Each tree in the forest is subsequently trained on a random subset of the training data. Each decision tree is grown depth-first in a greedy manner by choosing the feature and threshold for the decision stump so that class separation (expected gain in information) is maximised [58].

The `RDF` classifier has two key properties that motivate its use for edgel classification. First, it naturally handles multiple classes which is useful for separating edgels which describe inner and outer boundaries. Second, the depth-first greedy construction means that the decision trees automatically grow to handle classification over different sections of the object of interest. This is because the training data is split at each node

as the tree is grown so that not all features have to be simultaneously discriminative for every edgel. As a result, features (6.11) and (6.12) typically dominate the early levels of each decision tree, but features of the form of (6.10) are used at deeper levels to separate classification of different sections of the object.

6.5 Experiments

The purpose of this section is to examine the use of the `BFM` for the application of LV endocardium and epicardium boundary candidate selection from 3D echocardiography frames. The aim of the experiments is to understand how the parameters *specific to the* `BFM` impact detection accuracy and boundary candidate selection performance — complete optimisation and validation of *all* classifier components is therefore not covered.

6.5.1 Implementation

The `BFM` components were implemented in a combination of C and Python 2.7 with Numpy 1.6.1[4] and Scipy 0.10[5] extensions. The scikit-learn 0.10[6] machine learning library was also used, with extensions made to the `MeanShift` and `DecisionTreeClassifier` classes for object detection and handling of large matrices.

All experiments, except for the edgel classifier training, were run on machines with an Intel Xeon 5430 processor and 8GB of RAM. The edgel classifier training was done on a machine with 32GB of RAM.

6.5.2 Dataset

The dataset consisted of 275 3D echocardiography frames — 25 individual subjects with 11 frames each covering the cardiac cycle from end-diastole to end-systole. The images were acquired from a Philips iE33 ultrasound system with a mean image spatial resolution of $0.85 \times 0.86 \times 0.77\,\mathrm{mm}^3$ and standard dimensions of $224 \times 208 \times 208$. To assess the flexibility of the method to unseen subjects the dataset was split into three groups. Six of the subjects were used for constructing the `BFM`, six of the subjects were used to train the detection and edgel classifiers, and the remaining 13 subjects were used for testing.

[4] `http://www.numpy.org`
[5] `http://www.scipy.org`
[6] `http://scikit-learn.org/`

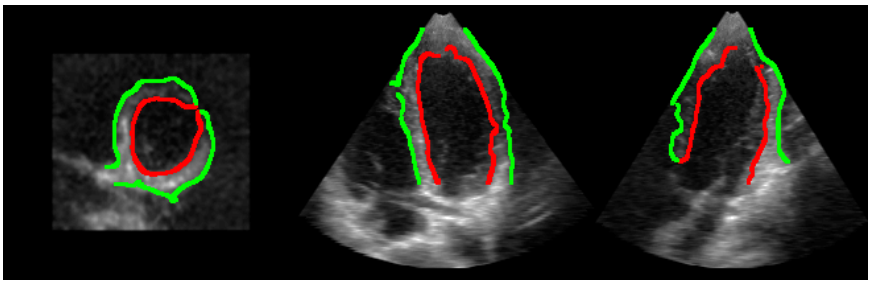


Figure 6.7: Echocardiogram slices with the inner/endocardium (red) and outer/epicardium (green) edgels labelled.

As a result of the anisotropic voxel spacing, each edgel's position for each edge part was defined in scale-normalised and *spacing-normalised* coordinates. The advantage of this formulation is that scale only refers to the size of the anatomical object of interest, and not the magnification at which the image was acquired. Boundary candidates were generated for each image using an isotropic log-Gabor filter with feature asymmetry[7] and non-maximum suppression (§3.4.2).

6.5.2.1 Labelled Edge Map Generation

Labelled edge maps of the training images were required to train the position and edgel classifiers (Figure 6.7). Manual segmentations of the myocardium were available for every image and these were used to label edgels which corresponded (roughly) to the endocardium and the epicardium. First, the manual segmentations were dilated in each short-axis slice and used as a mask to retain only the boundary candidates corresponding to the myocardium. Next, these edgels were classified as endocardium or epicardium depending on whether they were closer to the inner or outer edges of the myocardium segmentation.

6.5.3 Boundary Fragment Model Construction

There are eight parameters that must be set to construct the `BFM`: the number of fragments to sample (N), the minimum edgel density (ξ_{MIN}), the maximum edgel density (ξ_{MAX}), the expected scale variation (s_r), the upper distance limit (τ), the oriented chamfer matching weight (λ), the number of appearance clusters (K) and the positional clustering `MeanShift` bandwidth (σ). The first four parameters ($N, \xi_{\text{MIN}}, \xi_{\text{MAX}}, s_r$) affect the number and properties of individual fragments. The next two parameters

[7] $f_0 = 0.045\text{mm}^{-1}$, $b = 2$, and $T = 1.2$.

(τ, λ) determine *how* fragments are compared and for the construction stage can be set interactively. The final two parameters (K, σ) change how fragments are clustered.

6.5.3.1 Fragment Quality

A value of $N = 100 \times 66 = 6600$ was set so that 100 fragments (containing edgels from the endocardium and epicardium) were generated from each of the 66 source images. Note that N is only limited by the $\mathcal{O}(N^2)$ memory requirement for the distance matrix generated during the appearance clustering step (§6.2.5.2) and in general can be set as high as practicable. For ξ_{MIN} and ξ_{MAX}, it was found interactively that $\xi_{\text{MIN}} = 0.025$ and $\xi_{\text{MAX}} = 0.10$ were suitable settings to reject fragments that were too sparse/dense and not representative of local structure. $s_r = 1.1$ was set because scale variation was already present in the source images (scale change is present over the entire cardiac cycle).

6.5.3.2 Distance Specification

The parameters τ and λ control $d^{E,\lambda}_{\text{OCM}}$ and the definition of "close appearance" between two sets of edgels T and E. Increasing τ *decreases* the sensitivity of $d^{E,\lambda}_{\text{OCM}}$ to changes in displacement between edgel sets T and E, while increasing λ *increases* the sensitivity of $d^{E,\lambda}_{\text{OCM}}$ to changes in orientation.

To set (τ, λ), representative sets of edgels (E) from the LV were taken and perturbed from their fragment origin by small translations and rotations (E'). $d^{E,\lambda}_{\text{OCM}}(E', \mathbf{0})$ was then calculated for various (τ, λ). $\tau = 15$mm and $\lambda = 0.4$ were chosen interactively so that small perturbations measured $d^{E,\lambda}_{\text{OCM}}(E', \mathbf{0}) \ll 1$.

6.5.3.3 Fragment Clustering

The parameters K and σ control the number and properties of the edge parts in the final `BFM`. To understand the effect of the clustering parameters, nine different `BFM`s were constructed using three values for K and σ (Table 6.1).

K	100	100	100	150	150	150	200	200	200
σ	5	10	15	5	10	15	5	10	15
$\lvert\mathcal{M}\rvert$	287	185	123	382	236	182	388	301	246

Table 6.1: The number of parts $(|\mathcal{M}|)$ in each `BFM` generated with $(K, \sigma) \in \{100, 150, 200\} \times \{5, 10, 15\}$.

Comparing the number of edge parts generated by each configuration illustrates some general properties about the clustering procedure. First, increasing K creates more appearance clusters describing positionally similar regions, resulting in more edge parts. However, K cannot be increased indefinitely because the final appearance clusters become too sparse, resulting in edge parts which poorly summarise sections of the myocardium. This is shown by the small increase in the number of fragments from $K = 150$ to $K = 200$ compared to the increase from $K = 100$ to $K = 150$. Second, increasing σ results in fewer edge parts generated from each appearance cluster because each edge part can represent more fragments from a given appearance cluster.

Given the small increase in the number of fragments from $K = 150$ to $K = 200$, values of K greater than 200 were not investigated in the experiments that follow.

6.5.4 Left Ventricle Detection

Using the models with $\sigma = 5$ and $\sigma = 10$ only, centroid classifiers with $M \in \{100, 200, 300\}$ were trained. Part responses were calculated with $\lambda \in \{0.1, 0.4, 0.8\}$ to make up the feature vectors. For each of the 66 position classifier training frames, 153 candidate positions were evaluated around the centre of the object bounding box in scale-space to generate the classifier training data with $(\delta_1, \delta_2, \gamma_1, \gamma_2) = (0.5, 1.5, 1.1, 1.4)$. No additional negative training examples were included in the training data.

Detection was performed using a three-scale hierarchical grid structure where the centroid and scale at each level were determined using `MeanShift` with spatial bandwidth of 20mm and log-scale bandwidth of 0.2. These parameters were chosen to merge detection modes over similar positions and scales. The output of the detection step for each test frame was a centroid position and scale.

Two measures of performance were used to evaluate detection accuracy: the Euclidean distance between the detected centroid position and the ground-truth centroid, and the Jaccard index of the detected bounding box to the ground-truth bounding box. The detection bounding box was taken as the box which encompasses all edge parts which contributed positively to the detection result at their optimal offsets. The advantage of the Jaccard index is that it ignores position bias of the edge parts with respect to what is considered the centroid and implicitly assesses the overall scale accuracy. The disadvantage of the Jaccard index is that an incorrect edge part which positively supports the final centroid classifier can increase the bounding box and disproportionally decrease the Jaccard index. For this reason both measurements are reported in Figure 6.8 and Table 6.2 respectively.

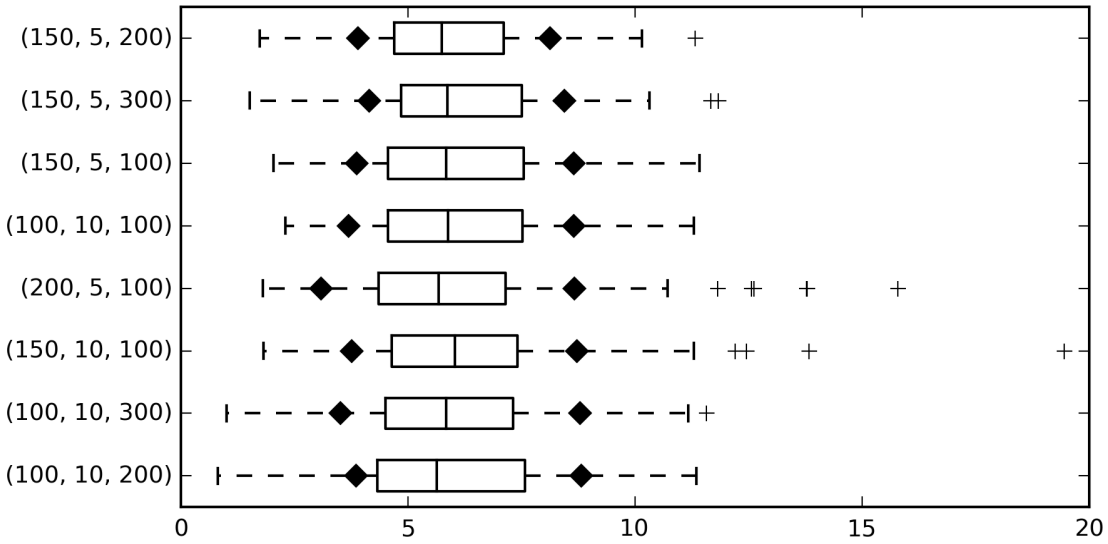


Figure 6.8: Position detection errors (mm) for the top 8 (out of 18) configurations based on the 90^{th} percentile (top to bottom). The 10^{th} and 90^{th} percentiles are marked by diamonds. Configuration labels are formatted as (K, σ, M).

K	σ	M	J_{10}	J_{50}	J_{90}	$\lvert J > 0.5 \rvert$
150	5	100	0.69	0.77	0.85	143
200	5	100	0.67	0.74	0.84	143
150	5	200	0.66	0.73	0.80	143
200	5	200	0.64	0.73	0.82	143
200	5	300	0.63	0.72	0.81	143
100	5	100	0.62	0.75	0.82	141
200	10	100	0.61	0.70	0.80	141
100	10	100	0.61	0.71	0.81	140

Table 6.2: Measurements of detection accuracy using the Jaccard index (J) for the top 8 (out of 18) configurations based on the 10^{th} percentile. The 10^{th}, 50^{th} and 90^{th} percentiles of the test errors (J_{10}, J_{50}, and J_{90}) are shown. Detections are counted if $J > 0.5$ (right column), out of a total of 143 test images.

From both Figure 6.8 and Table 6.2 it can be seen that the performance of all generated classifiers was similar. The first five configurations in Table 6.2 achieved zero detection errors based on the bounding box overlap criteria and this localisation accuracy is further confirmed in Figure 6.8. Furthermore, the top four configurations in Figure 6.8 — which have the lowest 90^{th} percentile values — all have $K < 200$, supporting the intuition of the previous section that for a given N, larger K is *not* automatically beneficial.

For completeness, the first five edge parts used by the centroid classifier $(150, 5, 100)$

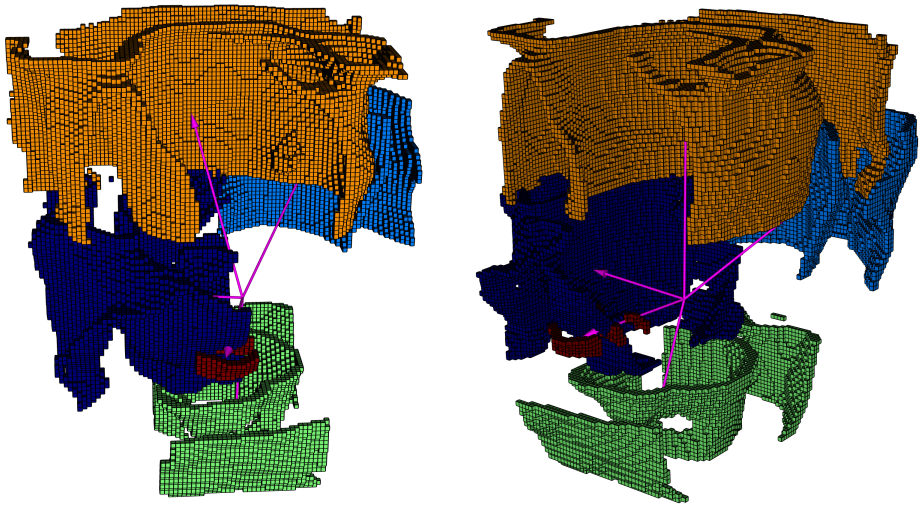


Figure 6.9: Two views of five edge parts used by the detection classifier $(150, 5, 100)$. (Shown at fixed scale and spacing, with the LV apex at the bottom.)

are shown in Figure 6.9. The choice of edge parts shows which sections of the myocardium are searched to determine support for the test centroid position and scale. Support near the apex is searched first, followed by support in the short-axis and then near the mitral valve.

6.5.5 Endocardium and Epicardium Delineation

To generate the edgel classifier training data, the centroid classifier $(150, 5, 100)$ was reapplied to each of the classifier training images to determine the centroid position and scale. Next, each part response was re-calculated to determine the optimal offset $\boldsymbol{x}_P$. The chamfer distance (6.7) and orientation distance (6.9) were calculated between every edgel $\underline{q}$ and every part $\underline{P}$. Each edgel which had a chamfer distance greater than 1.0 to every part was automatically classed as background and excluded from the training data. For the remaining edgels, the features were computed using (6.10), (6.11) and (6.12) with $\lambda \in \{0.4, 0.8\}$ to generate the feature vectors. The labels were determined from the labelled edge maps. Across all 66 classifier training images approximately 14 million feature vectors, each with ~ 770 elements were collected to generate the feature matrix.

6.5.5.1 Decision Tree Training and Interpretation

Eight decision trees were used to construct the edgel classifier. The training parameters were a maximum depth of 24 with a minimum of 10000 data points required to make

a split at a node. For all decision trees for configuration $(150, 5, 100)$, the first feature chosen was the minimum part distance to any edge part in the model (6.11) to remove background edgels. Next, orientation (6.12) was used to separate the endocardium and epicardium edgels. Distances to individual part responses (6.10) were subsequently used to split the classification of edgels over different sections of the model. The features (6.11) and (6.12) were then reused in subsequent splits because they describe distances to parts and the orientation *only* over the local section.

Importantly, the use of individual part responses as features means that a high maximum decision tree depth is required because when an individual part response is used it only affects a fraction of the training data. This results in an unbalanced division of the training data, which is not a problem, but further splits on different individual part responses are therefore required to handle other sections of the object.

Example classification results are shown in Figure 6.10.

The precision P and recall R measurements for myocardium/background and epicardium/endocardium classification were calculated for each test image and the F-ratio ($F = \frac{2PR}{P+R}$) was used to identify the three most difficult frames. The frames were from two different subjects and were all near end-systolic frames with the LV either out of frame or sections of the myocardium missing. Example slices from these frames are shown in Figure 6.11.

The difficulty with these frames is the large number of artefacts which are close to the endocardium. These artefacts are not always classified as background because they exhibit much of the same structure as the true sections of the myocardium over large reasons. This is a deficiency of the boundary candidate classifier because it only takes appearance context into account; it does not consider adjacent labelling or other structures. Despite this, the boundary candidates which describe the general structure of the endocardium and epicardium are still identified. For comparison, Figure 6.12 shows renderings of the endocardium-labelled edgels of the left-most frames from Figures 6.10 and 6.11[8].

6.5.6 Execution Times

Computing the full edge map for each test image took approximately 20 seconds and was dominated by the construction of the log-Gabor and three Riesz transform filters. Approximately 4 seconds was required to compute the full distance transform. For the chosen fragment model configuration, $(150, 5, 100)$, 60 seconds was required to locate

[8]For clarity, only the largest oriented connected components are shown.

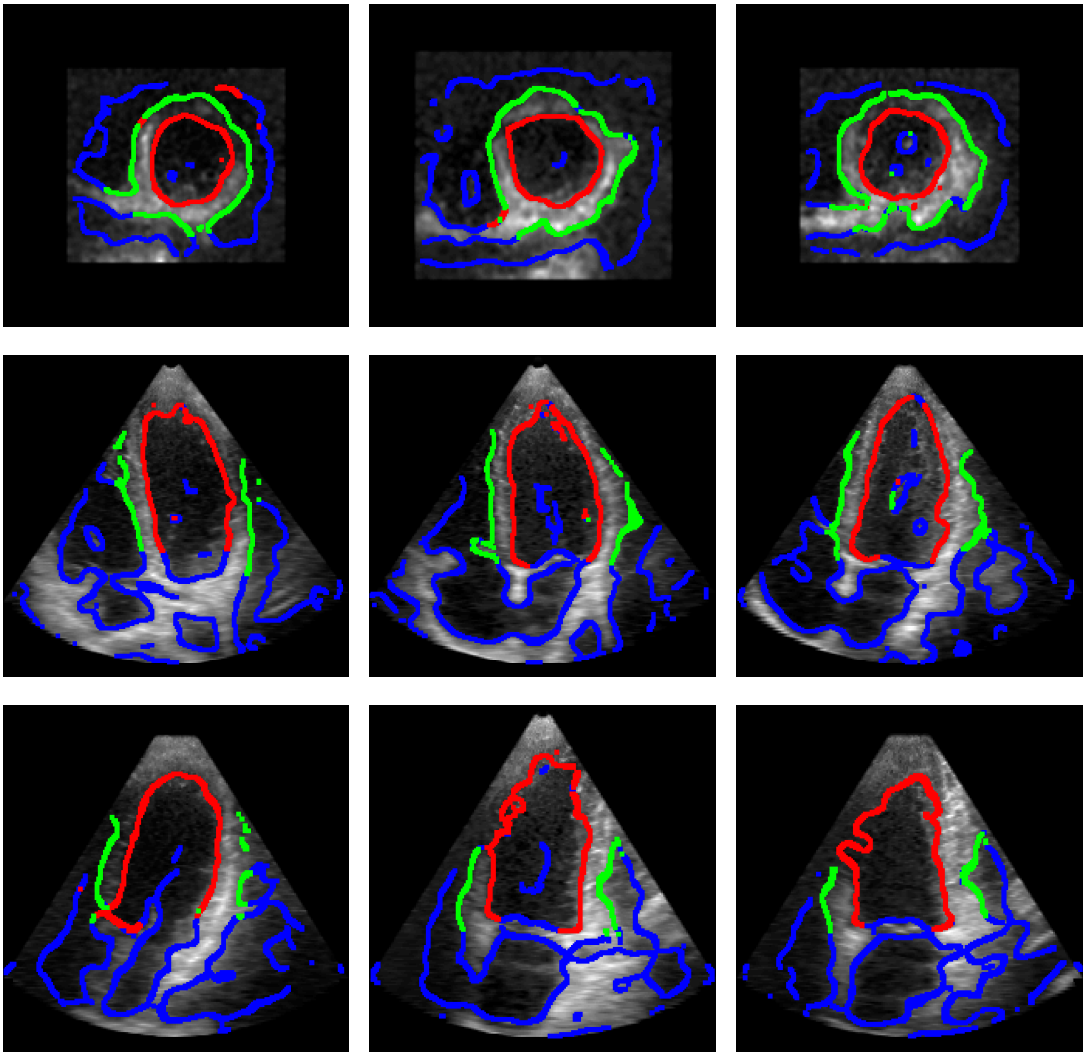


Figure 6.10: Short-axis and long-axis slices showing the endocardium (red), epicardium (green), and background (blue) boundary candidate classification results for three different subjects.

the centroid and scale of the LV using the three scale hierarchy. Finally, all steps for edgel classification took an additional 35 seconds. This time was dominated by the calculation of all distance transforms for each edge part.

6.6 Conclusions

In this chapter, a boundary candidate selection algorithm was described which does *not* require an initialisation or estimate of the model surface. The algorithm, built around the Boundary Fragment Model (BFM) was extended to object detection and boundary candidate selection in 3D. The application of left ventricle (LV) endocardium

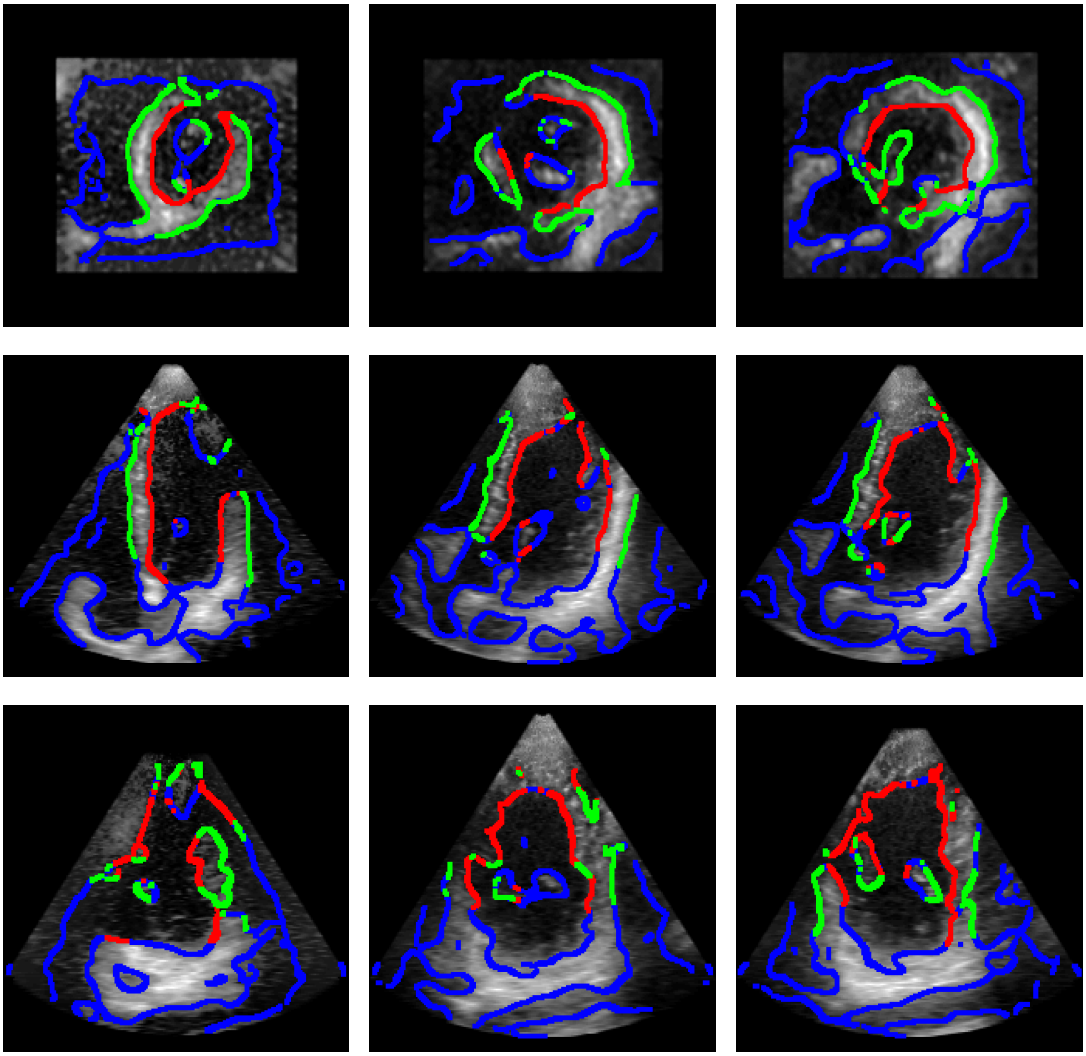


Figure 6.11: Short-axis and long-axis slices showing the endocardium (red), epicardium (green), and background (blue) boundary candidate classification results for the three most difficult frames (based on the F-ratio).

and epicardium delineation from 3D echocardiography frames was used to demonstrate the operation of the algorithm and understand the impact of key parameter choices. Importantly, the framework was shown to successfully detect the position and scale of the LV, and subsequently classify the boundary candidates as background, endocardium, or epicardium.

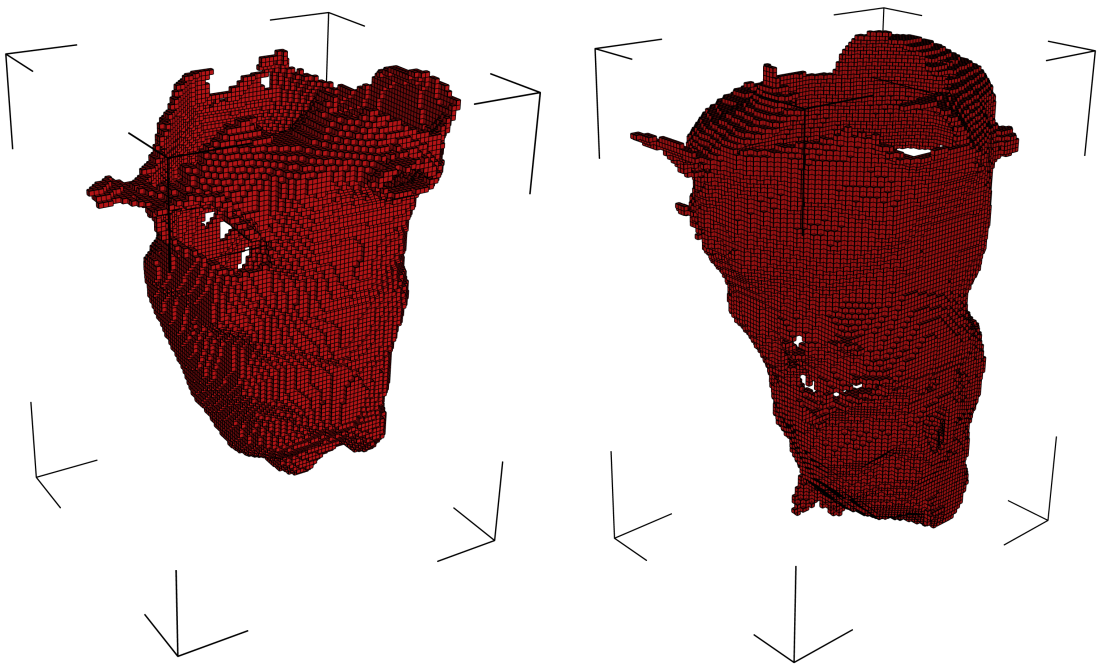


Figure 6.12: Renders of the endocardium-labelled edgels corresponding to the left-most frames from Figure 6.10 (left) and Figure 6.11 (right).

Chapter 7

Conclusions

In this thesis, extensions to model-based segmentation algorithms for the analysis of 2D and 3D ultrasound images and sequences have been presented. Using explicit model representations — B-splines and subdivision surfaces — and image-derived (geometric) model fitting, the algorithms necessary for *joint* optimisation of the model geometry and data point correspondences (footprints) have been described and demonstrated in full.

In Chapter 3, it has been shown in 2D that joint optimisation improves segmentation accuracy and facilitates boundary candidate selection algorithms which are more powerful than perpendicular search. Furthermore, it has been shown that this enables the specification of 2D+t segmentation algorithms which recover anatomically consistent model contours and segment all frames simultaneously (not sequentially).

In Chapter 4, the algorithm components are extended to 3D and demonstrated for three ultrasound image analysis problems: skull segmentation for fetal brain image analysis; fetal face segmentation for shape analysis, and left ventricle segmentation for volume measurement in echocardiography images. As a further extension, in Chapter 5 a framework to perform model-based segmentation of multiple 3D+t sequences, while jointly optimising an underlying linear basis shape model, has been presented for the challenging task of right ventricle segmentation from echocardiography images.

Finally, as a further contribution towards automatic segmentation, in Chapter 6 a framework to select boundary candidates independent of a model surface has been presented. The proposed method isolates relevant anatomical boundary positions in an image using only the structure of edges, and is built around a weak parts-based shape model — the Boundary Fragment Model (`BFM`). The task of identifying boundary candidates corresponding to the left ventricle endocardium and epicardium from 3D echocardiography images has been completely examined.

Although motivated by challenges in ultrasound image analysis, the conceptual contributions of this thesis are general and applicable to model-based segmentation problems in many domains. Moreover, the components are modular, enabling straightforward construction of application-specific formulations for new clinical problems as they arise in the future.

7.1 Future Work

With reference to Chapter 3, quantitative validation of anatomical consistency for `SEPARATE` and `SHARED` on 2D+t echocardiography sequences is ultimately desirable but requires additional phantom or multi-modal data. The benefit of this assessment would be to validate and improve the formulation of the appearance model $\mathcal{A}$ — the improved accuracy and robustness to initialisation of `SEPARATE` and `SHARED` in comparison to `SEQUENTIAL-FIXED` ("track-to-last") has already been demonstrated sufficiently by simulation (§3.4.4). Following this, given the model surface definitions in Chapter 4 and the algorithm for updating model correspondences, extending the 2D+t algorithms to 3D+t *is* straightforward, and validation should therefore be pursued as clinical interest develops and data becomes available. A multiple sequence 3D+t extension to Chapter 5 is also simple to formulate, and given the initial validation, could facilitate large-scale statistical analysis of detailed endocardium shape dynamics more refined than just changes in volume. Application to non-ultrasound and multi-modal segmentation problems (e.g. ultrasound-MR) should also be possible with few changes required.

Alongside this, to facilitate wider adoption and application of the methods presented in this thesis, it is desirable to reduce the execution time of the discrete and continuous optimisation algorithms used for boundary candidate selection and model fitting respectively. With reference to Chapter 3, for `PAIRWISE` and `SEPARATE`/`SHARED` with SSD for $\mathcal{A}$, one approach to improve the speed of the discrete optimisation step is to utilise the squared residual definitions of θ_{UNARY} and θ_{PAIRWISE}. An initial extension to the `MIN-SUM` algorithm — similar to that proposed by Amberg and Vetter [4] for optimal feature tracking — has been developed and implemented which, for each node, explores labels in a specific order, enabling a lower bound on the energy to be evaluated and early termination of the algorithm. Further development of this algorithm will be particularly important for any proposed 3D+t extensions because the increased label space size (number of boundary candidates) makes boundary candidate selection more difficult.

Finally, it is also desirable to achieve fully automatic segmentation where (a) boundary candidate positions *and* (b) boundary candidate correspondences are initialised *without* a model surface estimate. The proposed `BFM` framework (Chapter 6) achieves (a) by selecting relevant boundary candidates detected from a simple edge filter. The Structured Random Forest framework used in §4.4.3 achieves (a) by regressing, from the image, the probability that a given pixel belongs to the boundary of interest [46]. However, neither algorithm fulfils (b). Instead, a framework similar to the *Vitruvian Manifold* [149], which regresses data point correspondences from the input image, should be pursued to complement the model-based algorithms presented in this thesis.

Appendix A

Loop and Doo-Sabin Subdivision Details

A.1 Loop Subdivision

The purpose of this section is threefold. First, to provide evaluation and derivation details pertaining to ordinary and extraordinary Loop subdivision patches that are not readily available or accessible in existing literature (§A.1.1–§A.1.4). Second, to demonstrate *why*, for extraordinary patches, first and second derivatives with respect to local patch coordinates can vanish or be unbounded (§A.1.5–§A.1.7). Third, to provide implementation details for replacing extraordinary Loop patches with approximate triangle Bezier patches (§A.1.8, §A.1.9).

A.1.1 Triangle B-spline Basis Functions

Assume the vertex labelling of Figure 4.2a. Using the basis function definitions from [141], and making the substitution $u = 1 - v - w$ and subsequent variable changes

$v \to s$ and $w \to t$, the triangle B-spline basis functions are given by the rows of:

$$\frac{1}{12}\begin{bmatrix} -s^4-2s^3t+8s^3+12s^2t-12s^2-2st^3+12st^2-12st-t^4+8t^3-12t^2+6 \\ -s^4-2s^3t-4s^3-6s^2t+6s^2+4st^3-12st^2+6st+4s+2t^4-4t^3+2t+1 \\ 2s^4+4s^3t-4s^3-2st^3+6st^2-6st+2s-t^4+2t^3-2t+1 \\ -s^4-2s^3t+2s^3+2st^3-6st^2+6st-2s+t^4-4t^3+6t^2-4t+1 \\ s^4+2s^3t-4s^3-6s^2t+6s^2-2st^3+6st-4s-t^4+2t^3-2t+1 \\ -s^4-2s^3t+2s^3+6s^2t+4st^3-6st-2s+2t^4-4t^3+2t+1 \\ 2s^4+4s^3t-4s^3-12s^2t-2st^3-6st^2+6st+2s-t^4-4t^3+6t^2+4t+1 \\ -s^4-2s^3t+2s^3+6s^2t-2st^3+6st^2-t^4+2t^3 \\ s^4+2s^3t \\ -s^4-2s^3t+2s^3 \\ 2st^3+t^4 \\ -2st^3-t^4+2t^3 \end{bmatrix} \tag{A.1}$$

A.1.2 The Extended Subdivision Matrix

The definitions for A (the *"extended"* subdivision matrix) and $\bar{A}$ (the *"bigger"* subdivision matrix) are given precisely in [141]. However, since A specifically is important in the analysis that follows, its definition is included here for completeness.

The extended subdivision matrix $A \in \mathbb{R}^{(N+6)\times(N+6)}$ is given by:

$$A = \begin{bmatrix} S & 0 \\ S_{11} & S_{12} \end{bmatrix}$$

where $S_{11} \in \mathbb{R}^{5\times(N+1)}$ and $S_{12} \in \mathbb{R}^{5\times 5}$ are given by:

$$S_{11} = \tfrac{1}{16}\begin{bmatrix} 2 & 6 & 0 & 0 & \cdots 0 & 0 & 6 \\ 1 & 10 & 1 & 0 & \cdots 0 & 0 & 1 \\ 2 & 6 & 6 & 0 & \cdots 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & \cdots 0 & 1 & 10 \\ 2 & 0 & 0 & 0 & \cdots 0 & 6 & 6 \end{bmatrix} \quad S_{12} = \tfrac{1}{16}\begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 2 \end{bmatrix}$$

and $S \in \mathbb{R}^{(N+1)\times(N+1)}$ is the upper left subdivision matrix that contains the "extraordinary rules" of Loop's scheme:

$$S = \begin{bmatrix} a_N & b_N & b_N & b_N & b_N & \cdots & b_N & b_N & b_N \\ c & c & d & 0 & 0 & \cdots & 0 & 0 & d \\ c & d & c & d & 0 & \cdots & 0 & 0 & 0 \\ & & \vdots & & & \ddots & & \vdots & \\ c & d & 0 & 0 & 0 & \cdots & 0 & d & c \end{bmatrix} \tag{A.2}$$

with $a_N = 1 - \alpha(N)$, $b_N = \alpha(N)/N$, $c = 3/8$, $d = 1/8$, and:

$$\alpha(N) = \frac{5}{8} - \frac{(3 + 2\cos(2\pi/N))^2}{64}$$

A.1.3 Fourier Analysis of the Subdivision Matrix

The eigenstructure for A is derived in [141]. First, the eigenstructure of S (A.2), the upper left $(N+1) \times (N+1)$ block of A, is derived using Fourier analysis. The exact details of the analysis, which are omitted in [141], are included here for completeness.

For analysis of S it is sufficient to consider its application to a (column) vector of scalars. Let $\boldsymbol{x} = [\hat{x}\ x_0\ x_1 \cdots\ x_{N-1}]^\top \in \mathbb{R}^{N+1}$, where $\hat{x}$ is the (1D) extraordinary vertex position, and $\{x_i\}_{i=0}^{N-1}$ are the adjacent vertex positions. Furthermore, let $\boldsymbol{y} = [\hat{y}\ y_0\ y_1 \cdots\ y_{N-1}]^\top \in \mathbb{R}^{N+1}$ denote the vector of corresponding subdivided vertex positions, i.e. $\boldsymbol{y} = S\boldsymbol{x}$. The definitions for $\hat{y}$ and y_i are then given separately:

$$\hat{y} = a_N \hat{x} + b_N \sum_{i=0}^{N-1} x_i \tag{A.3}$$

$$y_i = c\hat{x} + cx_i + dx_{i\oplus 1} + dx_{i\ominus 1} \tag{A.4}$$

where $i \oplus 1 \triangleq (i+1) \mod N$ and $i \ominus 1 \triangleq (i-1) \mod N$.

The standard Discrete Fourier Transform (DFT) of a sequence x_i, denoted by $\underline{x}_k$, and its inverse, are given by [101, pp. 234–236]:

$$\underline{x}_k = \sum_{i=0}^{N-1} x_i e^{-j2\pi ik/N} \ \Leftrightarrow\ x_i = \frac{1}{N} \sum_{k=0}^{N-1} \underline{x}_k e^{j2\pi ik/N}$$

The DFT used in this section, to be consistent with [141], normalises $\underline{x}_k$ instead of x_k:

$$\underline{x}_k = \frac{1}{N} \sum_{i=0}^{N-1} x_i e^{-j2\pi ik/N} \ \overset{\text{DFT}}{\Leftrightarrow}\ x_i = \sum_{k=0}^{N-1} \underline{x}_k e^{j2\pi ik/N}$$

Treating x_i, $\hat{x}$, y_i, and $\hat{y}$ as distinct discrete sequences, (A.3) and (A.4) become:

$$\begin{aligned} \underline{\hat{y}} &= a_N \underline{\hat{x}} + N b_N \underline{x}_0 \\ \underline{y}_k &= c\delta(k)\, \underline{\hat{x}} + c\underline{x}_k + d\underline{x}_k e^{-j2\pi k/N} + d\underline{x}_k e^{j2\pi k/N} \end{aligned} \tag{A.5}$$

where $\delta(k)$ is 1 for $k = 0$, and 0 otherwise. For $k = 0$, (A.5) simplifies to:

$$\underline{y}_0 = c\underline{\hat{x}} + (c + 2d)\, \underline{x}_0$$

and for $k > 0$:

$$\begin{aligned} \underline{y}_k &= c\underline{x}_k + d\underline{x}_k e^{-j2\pi k/N} + d\underline{x}_k e^{j2\pi k/N} \\ &= c\underline{x}_k + 2d\cos(2\pi k/N) \\ &= \frac{3}{8} + \frac{2}{8}\cos(2\pi k/N) = f(k) \end{aligned}$$

which in matrix form is:

$$\begin{bmatrix} \underline{\hat{y}} \\ \underline{y}_0 \\ \vdots \\ \underline{y}_{N-1} \end{bmatrix} = \underbrace{\begin{bmatrix} a_N & Nb_N & 0 & 0 & \cdots & 0 \\ c & c+2d & 0 & 0 & \cdots & 0 \\ 0 & 0 & f(1) & 0 & \cdots & 0 \\ 0 & 0 & 0 & f(2) & \cdots & 0 \\ & & \cdots & & \ddots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & f(N-1) \end{bmatrix}}_{=\underline{S}} \begin{bmatrix} \underline{\hat{x}} \\ \underline{x}_0 \\ \vdots \\ \underline{x}_{N-1} \end{bmatrix}$$

where $\underline{S}$ is identical to $\hat{\boldsymbol{S}}$ in [141].

A.1.4 Thin-Plate Quadratic Form

The thin-plate energy for a single ordinary or approximate Loop patch is given by:

$$E_{\mathrm{TP}}(X) = \int_{\boldsymbol{t}\in\triangle} \|\chi_{xx}(\boldsymbol{t}, X)\|^2 + 2\|\chi_{xy}(\boldsymbol{t}, X)\|^2 + \|\chi_{yy}(\boldsymbol{t}, X)\|^2 \, \mathrm{d}\boldsymbol{t}$$

where X is the matrix of patch control vertices (row vectors). The second derivatives of χ are with respect to x and y, where $x = s + \frac{1}{2}t$ and $y = \frac{\sqrt{3}}{2}t$, *but* are functions in $\boldsymbol{t}$. The reparameterisation prior to differentiating transforms the local patch domain from a right angled unit triangle to a symmetric equilateral triangle with sides of unit length. This ensures the thin-plate formulation is isotropic.

Let $\boldsymbol{y} = [x\ y]^\top$ so that:

$$\boldsymbol{y} = \begin{bmatrix} 1 & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} \end{bmatrix} \boldsymbol{t} \Leftrightarrow \boldsymbol{t} = \begin{bmatrix} 1 & -\frac{1}{\sqrt{3}} \\ 0 & \frac{2}{\sqrt{3}} \end{bmatrix} \boldsymbol{y} \tag{A.6}$$

and making the change of variables gives:

$$E_{\mathrm{TP}}(X) = \frac{2}{\sqrt{3}} \int_{y=0}^{\frac{\sqrt{3}}{2}} \int_{x=\frac{1}{\sqrt{3}}y}^{1-\frac{1}{\sqrt{3}}y} \|\chi_{xx}(\boldsymbol{y}, X)\|^2 + 2\|\chi_{xy}(\boldsymbol{y}, X)\|^2 + \|\chi_{yy}(\boldsymbol{y}, X)\|^2 \, \mathrm{d}x \, \mathrm{d}y$$

For a single ordinary triangle B-spline patch, the second derivatives, which are functions in $\boldsymbol{y}$, are straightforward to formulate — using the triangle B-spline basis functions (A.1), substitute s and t for x and y (A.6), and differentiate twice. Furthermore, the integrand is a polynomial in x and y and capable of being integrated using `Sympy`[1].

[1] `http://www.sympy.org`

Assuming row-major $X \in \mathbb{R}^{12\times 3}$, let $\bar{\boldsymbol{x}} \in \mathbb{R}^{36}$ denote the column-wise flattened X. The evaluated integral in quadratic form is then given by:

$$E_{\mathrm{TP}}(X) = \bar{\boldsymbol{x}}^{\top} \begin{bmatrix} M & 0 & 0 \\ 0 & M & 0 \\ 0 & 0 & M \end{bmatrix} \bar{\boldsymbol{x}} = \left\| \begin{bmatrix} \sqrt{M} & 0 & 0 \\ 0 & \sqrt{M} & 0 \\ 0 & 0 & \sqrt{M} \end{bmatrix} \bar{\boldsymbol{x}} \right\|^2$$

where $M \in \mathbb{R}^{12\times 12}$ is:

$$M = \frac{-1}{162} \times \begin{bmatrix} -114 & 6 & 33 & 9 & 9 & 33 & 6 & -6 & -3 & 15 & -3 & 15 \\ 6 & -114 & 33 & 15 & -3 & -6 & 6 & 33 & 9 & 9 & 15 & -3 \\ 33 & 33 & -42 & -3 & 6 & -3 & -6 & -3 & 6 & -3 & -9 & -9 \\ 9 & 15 & -3 & -10 & -2 & 6 & -3 & -9 & -1 & 1 & -2 & -1 \\ 9 & -3 & 6 & -2 & -10 & -3 & 15 & -9 & -2 & -1 & -1 & 1 \\ 33 & -6 & -3 & 6 & -3 & -42 & 33 & -3 & -9 & -9 & 6 & -3 \\ 6 & 6 & -6 & -3 & 15 & 33 & -114 & 33 & 15 & -3 & 9 & 9 \\ -6 & 33 & -3 & -9 & -9 & -3 & 33 & -42 & -3 & 6 & -3 & 6 \\ -3 & 9 & 6 & -1 & -2 & -9 & 15 & -3 & -10 & -2 & 1 & -1 \\ 15 & 9 & -3 & 1 & -1 & -9 & -3 & 6 & -2 & -10 & -1 & -2 \\ -3 & 15 & -9 & -2 & -1 & 6 & 9 & -3 & 1 & -1 & -10 & -2 \\ 15 & -3 & -9 & -1 & 1 & -3 & 9 & 6 & -1 & -2 & -2 & -10 \end{bmatrix}$$

Definitions for M for extraordinary patches approximated by triangle Bezier patches (§A.1.9) can be found by the same procedure.

A.1.5 Evaluating Extraordinary Patches at the Origin

Consider an extraordinary patch with an extraordinary vertex of valency $N \notin \{3, 6\}$. For now, let $X \in \mathbb{R}^{(N+6)\times 3}$ denote the matrix of positions, as row vectors, of the $N+6$ patch control vertices. Following (4.2) and [141], the closed-form of χ is:

$$\chi(\boldsymbol{t}, X) = \boldsymbol{b}\left(t_{k,n}(\boldsymbol{t})\right)^{\top} P_k \bar{A} A^{n-1} X \tag{A.7}$$

where the descriptions of all constituent functions and subdivision matrices from §4.1.1 apply.

To evaluate $\chi(\mathbf{0}, X)$, consider the limit $\boldsymbol{t} \to \mathbf{0}$. Let $\boldsymbol{t} = \boldsymbol{\epsilon}$, so that $\|\boldsymbol{\epsilon}\| \ll 1$ and $n \gg 1$. Also, let $V = [\boldsymbol{v}_i]$ denote the matrix of right-eigenvectors of A, $\tilde{V} = [\tilde{\boldsymbol{v}}_i] = \left(V^{-1}\right)^{\top}$ denote the matrix of left-eigenvectors, and $\Lambda = \mathrm{diag}([\lambda_i])$ denote the diagonal

matrix of (sorted) eigenvalues (refer to [141] for the complete definitions). Then:

$$\begin{aligned}
\chi\left(\boldsymbol{\epsilon}, X\right) &= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} A^{n-1} X \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(V \Lambda \tilde{V}^{\top}\right)^{n-1} X \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(V \Lambda^{n-1} \tilde{V}^{\top}\right) X \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\sum_{i=1}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i^{\top}\right) X \\
&\approx \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\lambda_1^{n-1} \boldsymbol{v}_1 \tilde{\boldsymbol{v}}_1^{\top}\right) X \qquad \text{(A.8)} \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\mathbf{1} \tilde{\boldsymbol{v}}_1^{\top}\right) X \qquad \text{(A.9)} \\
&= \tilde{\boldsymbol{v}}_1^{\top} X \qquad \text{(A.10)}
\end{aligned}$$

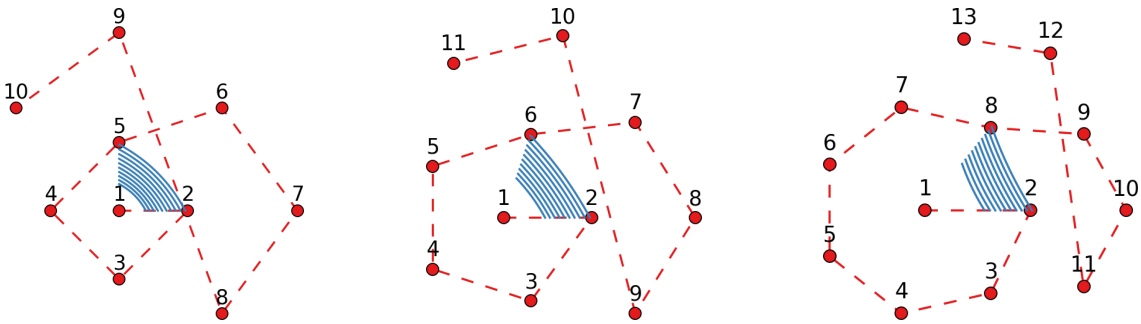
where (A.8) follows because $\lambda_1 = 1 > \lambda_2 \geq \cdots \geq \lambda_{N+6}$, (A.9) follows because $\boldsymbol{v}_1 = \mathbf{1}$, and (A.10) follows because the entries of $\boldsymbol{b}$ and rows of $\bar{A}$ sum to one. Importantly, (A.10) is also true for $N = 3$ as the eigenvalue of the Jordan block is less than one.

A.1.6 The Characteristic Map

For an extraordinary patch with valency $N \notin \{3, 6\}$, consider evaluating χ *near* $\boldsymbol{t} = \mathbf{0}$. Let $\boldsymbol{t} = \boldsymbol{\epsilon}$, where $\|\boldsymbol{\epsilon}\| \ll 1$ so that $n \gg 1$. Now:

$$\begin{aligned}
\chi\left(\boldsymbol{\epsilon}, X\right) - \chi\left(\mathbf{0}, X\right) &= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} A^{n-1} X - \underbrace{\tilde{\boldsymbol{v}}_1^{\top} X}_{\chi(\mathbf{0}, X)} \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\sum_{i=1}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i^{\top}\right) X - \tilde{\boldsymbol{v}}_1^{\top} X \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\sum_{i=1}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i^{\top} - \mathbf{1} \tilde{\boldsymbol{v}}_1^{\top}\right) X \qquad \text{(A.11)} \\
&= \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\sum_{i=2}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i^{\top}\right) X \qquad \text{(A.12)} \\
&\approx \boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \left(\lambda_2^{n-1} \boldsymbol{v}_2 \tilde{\boldsymbol{v}}_2^{\top} + \lambda_3^{n-1} \boldsymbol{v}_3 \tilde{\boldsymbol{v}}_3^{\top}\right) X \qquad \text{(A.13)} \\
&= \lambda_2^{n-1} \underbrace{\boldsymbol{b}\left(t_{k,n}\left(\boldsymbol{\epsilon}\right)\right)^{\top} P_k \bar{A} \begin{bmatrix} \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix}}_{\text{A}} \underbrace{\begin{bmatrix} \tilde{\boldsymbol{v}}_2^{\top} \\ \tilde{\boldsymbol{v}}_3^{\top} \end{bmatrix} X}_{\text{B}} \qquad \text{(A.14)}
\end{aligned}$$

where (A.11) follows because the entries of $\boldsymbol{b}$ and rows of $\bar{A}$ sum to one, (A.12) follows because $\lambda_1 = 1$ and $\boldsymbol{v}_1 = \mathbf{1}$, and (A.13) and (A.14) follow because $\lambda_2 = \lambda_3 > \lambda_4 \geq \cdots \geq \lambda_{N+6}$.



(a) $N = 4$ (b) $N = 5$ (c) $N = 7$

Figure A.1: Example $[\boldsymbol{v}_2\ \boldsymbol{v}_3]$ (red) and evaluated characteristic maps (blue) for $N \in \{4, 5, 7\}$.

For similar $\boldsymbol{\epsilon}$, the terms in A of (A.14) evaluate to points on a 2D plane defined by the matrix of control points $[\boldsymbol{v}_2\ \boldsymbol{v}_3] \in \mathbb{R}^{(N+6)\times 2}$. This is known as the *characteristic map* [132] (Figure A.1). Importantly, the terms in B transform this plane into 3D based on X. Specifically, $\tilde{\boldsymbol{v}}_2{}^\top X$ and $\tilde{\boldsymbol{v}}_3{}^\top X$ span the tangent plane of the extraordinary patch in the limit as $\boldsymbol{t} \to \mathbf{0}$.

A.1.7 Degenerate First and Second Derivatives of Extraordinary Patches

For an extraordinary patch with valency $N \notin \{3, 6\}$, consider evaluating $\chi_s = \partial\chi/\partial s$ near $\boldsymbol{t} = (s, t) = \mathbf{0}$. Define $\boldsymbol{\epsilon} = (\epsilon, 0)$ and assume $\|\boldsymbol{\epsilon}\| \ll 1$ so that $n \gg 1$. Using (A.12), the derivative of $\chi(\boldsymbol{t}, X) - \chi(\mathbf{0}, X)$ with respect to s at $\boldsymbol{t} = \boldsymbol{\epsilon}$, is:

$$
\begin{aligned}
&\frac{\partial}{\partial s}\left\{\chi(\boldsymbol{t}, X) - \chi(\mathbf{0}, X)\right\}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \\
&= \frac{\partial}{\partial s}\left\{\boldsymbol{b}\left(t_{k,n}(\boldsymbol{t})\right)^\top P_k \bar{A}\left(\sum_{i=2}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X\right\}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \\
&= \frac{\partial t_{0,n}(\boldsymbol{t})}{\partial s}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \boldsymbol{b}_s\left(t_{0,n}(\boldsymbol{\epsilon})\right)^\top P_0 \bar{A}\left(\sum_{i=2}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\
&= 2^n \boldsymbol{b}_s\left(t_{0,n}(\boldsymbol{\epsilon})\right)^\top P_0 \bar{A}\left(\sum_{i=2}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \qquad \text{(A.15)} \\
&= 2\boldsymbol{b}_s\left(t_{0,n}(\boldsymbol{\epsilon})\right)^\top P_0 \bar{A}\left(\sum_{i=2}^{N+6} (2\lambda_i)^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\
&\approx (2\lambda_2)^{n-1}\left[2\boldsymbol{b}_s\left(t_{0,n}(\boldsymbol{\epsilon})\right)^\top P_0 \bar{A}\left(\boldsymbol{v}_2 \tilde{\boldsymbol{v}}_2{}^\top + \boldsymbol{v}_3 \tilde{\boldsymbol{v}}_3{}^\top\right) X\right] \qquad \text{(A.16)}
\end{aligned}
$$

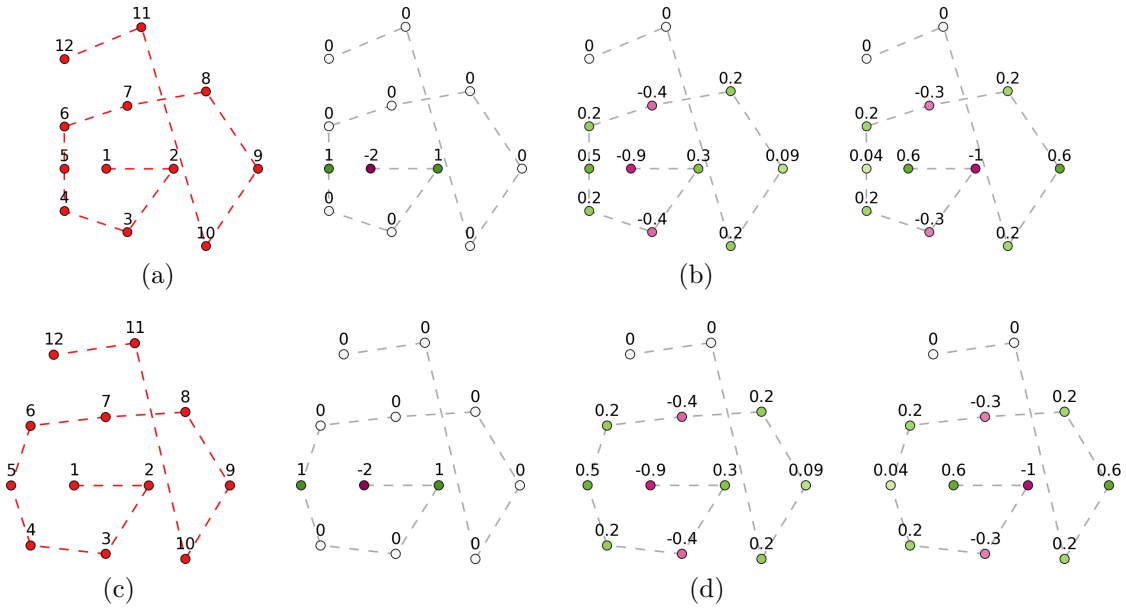


Figure A.2: (a) $N = 4$. Control points (red) for $P_0\bar{A}\left[\boldsymbol{v}_2\ \boldsymbol{v}_3\right]$. (b) Weights (purple to green) for $\boldsymbol{b}_{ss}\left(\boldsymbol{t}\right)$ where $\boldsymbol{t} = (s, 0)$ and $s \in \{0, 0.3, 0.8\}$. (c), (d) $N = 5$.

where $k = 0$ because $\boldsymbol{\epsilon} = (\epsilon, 0)$, and $\boldsymbol{b}_s$ is the vector of first derivatives with respect to s of the triangle B-spline basis functions. (A.15) follows from the definition of $t_{0,n}$, and (A.16) follows because $\lambda_2 = \lambda_3 > \lambda_4 \geq \cdots \geq \lambda_{N+6}$.

With reference to [141], for $3 \leq N < 6$, $\lambda_2 < \frac{1}{2}$ so that $\lim_{\boldsymbol{\epsilon}\to\mathbf{0}} \left\|\chi_s\left(\boldsymbol{\epsilon}, X\right)\right\| = 0$, i.e. the first derivative with respect to s vanishes as $\boldsymbol{t} \to \mathbf{0}$. (For $N = 3$, the single Jordan block at eigenvalue $\frac{1}{16}$ has no impact since $\lambda_2 > \frac{1}{16}$.)

For $N > 6$, $\lambda_2 > \frac{1}{2}$ and $\lim_{\boldsymbol{\epsilon}\to\mathbf{0}} \left\|\chi_s\left(\boldsymbol{\epsilon}, X\right)\right\| \to \infty$, i.e. the first derivative with respect to s is unbounded as $\boldsymbol{t} \to \mathbf{0}$. Similar results can be demonstrated for χ_t.

Now consider evaluating $\chi_{ss} = \partial\chi_s/\partial s$ for $\boldsymbol{t} = \boldsymbol{\epsilon}$. From (A.15), 2^n is replaced with 4^n and $\boldsymbol{b}_s$ is replaced with $\boldsymbol{b}_{ss}$:

$$\begin{aligned}\chi_{ss}\left(\boldsymbol{\epsilon}, X\right) &= 4^n \boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0\bar{A}\left(\sum_{i=2}^{N+6} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\ &= 4\boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0\bar{A}\left(\sum_{i=2}^{N+6} \left(4\lambda_i\right)^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X\end{aligned}$$

For $N = 3$, this is finite because $\lambda_2 = \lambda_3 = \frac{1}{4} > \lambda_4 \geq \cdots \geq \lambda_9$. For $N > 3$, the inner product of the control points $P_0\bar{A}\left[\boldsymbol{v}_2\ \boldsymbol{v}_3\right]$ (e.g. Figures A.2a, A.2c) and $\boldsymbol{b}_{ss}\left(\boldsymbol{t}\right)$ (e.g. Figures A.2b, A.2d) is non-zero for some $\boldsymbol{t} = (s, 0)$ so that χ_{ss} remains dependent on λ_2 and λ_3. Since $\lambda_2 = \lambda_3 > \frac{1}{4}$, it is unbounded.

A.1.8 Triangle Quartic Bezier Patch

Assume the vertex labelling of Figure 4.2a. Using the basis function definitions from [141], and making the variable changes $u \to r$, $v \to s$ and $w \to t$, the triangle *barycentric* B-spline basis functions are given by the rows of the vector $\breve{\boldsymbol{b}}(r,s,t)$:

$$\frac{1}{12}\begin{bmatrix} 6r^4+24r^3s+24r^3t+24r^2s^2+60r^2st+24r^2t^2+8rs^3+36rs^2t+\dots \\ \dots\ 36rst^2+8rt^3+s^4+6s^3t+12s^2t^2+6st^3+t^4 \\ r^4+8r^3s+6r^3t+24r^2s^2+36r^2st+12r^2t^2+24rs^3+60rs^2t+\dots \\ \dots\ 36rst^2+6rt^3+6s^4+24s^3t+24s^2t^2+8st^3+t^4 \\ r^4+6r^3s+2r^3t+12r^2s^2+6r^2st+6rs^3+6rs^2t+s^4+2s^3t \\ r^4+2r^3s \\ r^4+2r^3t \\ r^4+2r^3s+6r^3t+6r^2st+12r^2t^2+6rst^2+6rt^3+2st^3+t^4 \\ r^4+6r^3s+8r^3t+12r^2s^2+36r^2st+24r^2t^2+6rs^3+36rs^2t+\dots \\ \dots\ 60rst^2+24rt^3+s^4+8s^3t+24s^2t^2+24st^3+6t^4 \\ 2rs^3+6rs^2t+6rst^2+2rt^3+s^4+6s^3t+12s^2t^2+6st^3+t^4 \\ s^4+2s^3t \\ 2rs^3+s^4 \\ 2st^3+t^4 \\ 2rt^3+t^4 \end{bmatrix}$$

The *barycentric* position function is $\breve{\chi}(r,s,t,X) = \breve{\boldsymbol{b}}(r,s,t)^\top X = \boldsymbol{b}(s,t)^\top X$, where $X \in \mathbb{R}^{12\times 3}$ is the matrix of positions (row vectors) of the control vertices.

By inspecting the form of $\breve{\chi}(r,s,t,X)$ at its boundaries, the Bezier position function $\hat{\chi}$ for a regular triangle B-spline is derived:

$$\begin{aligned} \hat{\chi}(r,s,t,P) = {} & \boldsymbol{p}_0 s^4 + \boldsymbol{p}_1 t^4 + \boldsymbol{p}_2 r^4 \\ & + 4\boldsymbol{p}_0^+ s^3 t + 6\boldsymbol{p}_{01} s^2 t^2 + 4\boldsymbol{p}_1^- s t^3 \\ & + 4\boldsymbol{p}_1^+ t^3 r + 6\boldsymbol{p}_{12} t^2 r^2 + 4\boldsymbol{p}_2^- t r^3 \\ & + 4\boldsymbol{p}_2^+ r^3 s + 6\boldsymbol{p}_{20} r^2 s^2 + 4\boldsymbol{p}_0^- r s^3 \\ & + 12str(s\bar{\boldsymbol{p}}_0 + t\bar{\boldsymbol{p}}_1 + r\bar{\boldsymbol{p}}_2) \end{aligned} \tag{A.17}$$

where $P \in \mathbb{R}^{15\times 3}$ is the matrix of 15 Bezier coefficients (vertices) which are defined in

terms of the 12 control vertices by:

$$P = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_0^- \\ \mathbf{p}_0^+ \\ \mathbf{p}_1^- \\ \mathbf{p}_1^+ \\ \mathbf{p}_2^- \\ \mathbf{p}_2^+ \\ \mathbf{p}_{01} \\ \mathbf{p}_{12} \\ \mathbf{p}_{20} \\ \bar{\mathbf{p}}_0 \\ \bar{\mathbf{p}}_1 \\ \bar{\mathbf{p}}_2 \end{bmatrix} = \frac{1}{24} \begin{bmatrix} 2 & 12 & 2 & 0 & 0 & 0 & 2 & 2 & 2 & 2 & 0 & 0 \\ 2 & 2 & 0 & 0 & 0 & 2 & 12 & 2 & 0 & 0 & 2 & 2 \\ 12 & 2 & 2 & 2 & 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \\ 4 & 12 & 3 & 0 & 0 & 0 & 3 & 1 & 0 & 1 & 0 & 0 \\ 3 & 12 & 1 & 0 & 0 & 0 & 4 & 3 & 1 & 0 & 0 & 0 \\ 3 & 4 & 0 & 0 & 0 & 1 & 12 & 3 & 0 & 0 & 1 & 0 \\ 4 & 3 & 0 & 0 & 0 & 3 & 12 & 1 & 0 & 0 & 0 & 1 \\ 12 & 3 & 1 & 0 & 1 & 3 & 4 & 0 & 0 & 0 & 0 & 0 \\ 12 & 4 & 3 & 1 & 0 & 1 & 3 & 0 & 0 & 0 & 0 & 0 \\ 4 & 8 & 0 & 0 & 0 & 0 & 8 & 4 & 0 & 0 & 0 & 0 \\ 8 & 4 & 0 & 0 & 0 & 4 & 8 & 0 & 0 & 0 & 0 & 0 \\ 8 & 8 & 4 & 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 & 0 \\ 6 & 10 & 1 & 0 & 0 & 0 & 6 & 1 & 0 & 0 & 0 & 0 \\ 6 & 6 & 0 & 0 & 0 & 1 & 10 & 1 & 0 & 0 & 0 & 0 \\ 10 & 6 & 1 & 0 & 0 & 1 & 6 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} X \tag{A.18}$$

The Bezier form is useful because its coefficients can be defined in terms of evaluated

positions and derivatives of the barycentric position function[2]:

$$
\begin{aligned}
\boldsymbol{p}_0 &= \breve{\chi}\left(0,1,0\right) \\
\boldsymbol{p}_1 &= \breve{\chi}\left(0,0,1\right) \\
\boldsymbol{p}_2 &= \breve{\chi}\left(1,0,0\right) \\
\boldsymbol{p}_0^- &= \frac{\breve{\chi}_r\left(0,1,0\right) - \breve{\chi}_s\left(0,1,0\right)}{4} + \boldsymbol{p}_0 \\
\boldsymbol{p}_0^+ &= \frac{\breve{\chi}_t\left(0,1,0\right) - \breve{\chi}_s\left(0,1,0\right)}{4} + \boldsymbol{p}_0 \\
\boldsymbol{p}_1^- &= \frac{\breve{\chi}_s\left(0,0,1\right) - \breve{\chi}_t\left(0,0,1\right)}{4} + \boldsymbol{p}_1 \\
\boldsymbol{p}_1^+ &= \frac{\breve{\chi}_r\left(0,0,1\right) - \breve{\chi}_t\left(0,0,1\right)}{4} + \boldsymbol{p}_1 \\
\boldsymbol{p}_2^- &= \frac{\breve{\chi}_t\left(1,0,0\right) - \breve{\chi}_r\left(1,0,0\right)}{4} + \boldsymbol{p}_2 \\
\boldsymbol{p}_2^+ &= \frac{\breve{\chi}_s\left(1,0,0\right) - \breve{\chi}_r\left(1,0,0\right)}{4} + \boldsymbol{p}_2 \\
\boldsymbol{p}_{01} &= \frac{16\breve{\chi}\left(0,\frac{1}{2},\frac{1}{2}\right) - 4\left(\boldsymbol{p}_0^+ + \boldsymbol{p}_1^-\right) - \left(\boldsymbol{p}_0 + \boldsymbol{p}_1\right)}{6} \\
\boldsymbol{p}_{12} &= \frac{16\breve{\chi}\left(\frac{1}{2},0,\frac{1}{2}\right) - 4\left(\boldsymbol{p}_1^+ + \boldsymbol{p}_2^-\right) - \left(\boldsymbol{p}_1 + \boldsymbol{p}_2\right)}{6} \\
\boldsymbol{p}_{20} &= \frac{16\breve{\chi}\left(\frac{1}{2},\frac{1}{2},0\right) - 4\left(\boldsymbol{p}_2^+ + \boldsymbol{p}_0^-\right) - \left(\boldsymbol{p}_2 + \boldsymbol{p}_0\right)}{6} \\
\bar{\boldsymbol{p}}_0 &= \frac{-\boldsymbol{p}_0 + \boldsymbol{p}_1 + \boldsymbol{p}_2 - 4\left(\boldsymbol{p}_1^- + \boldsymbol{p}_1^+ + \boldsymbol{p}_2^- + \boldsymbol{p}_2^+\right) + 27\breve{\chi}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)}{12} \\
\bar{\boldsymbol{p}}_1 &= \frac{\boldsymbol{p}_0 - \boldsymbol{p}_1 + \boldsymbol{p}_2 - 4\left(\boldsymbol{p}_0^- + \boldsymbol{p}_0^+ + \boldsymbol{p}_2^- + \boldsymbol{p}_2^+\right) + 27\breve{\chi}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)}{12} \\
\bar{\boldsymbol{p}}_2 &= \frac{\boldsymbol{p}_0 + \boldsymbol{p}_1 - \boldsymbol{p}_2 - 4\left(\boldsymbol{p}_0^- + \boldsymbol{p}_0^+ + \boldsymbol{p}_1^- + \boldsymbol{p}_1^+\right) + 27\breve{\chi}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)}{12}
\end{aligned}
$$

In summary, the Bezier coefficients for *any* surface of the form (A.17) can be recovered with knowledge of *only* a finite number of evaluated positions and derivatives on the surface. If the points and derivatives have been acquired from an ordinary triangle B-spline, then the 15 Bezier coefficients are a linear combination of the original 12 control vertices (A.18).

A.1.9 Approximate Extraordinary Patches

Extraordinary patches have first and second derivative properties near the origin that are undesirable and detrimental to continuous optimisation (§A.1.7). To overcome

[2]The dependence of $\breve{\chi}$ on X has been omitted for clarity.

this, one possibility is to replace the extraordinary patch with a different surface model entirely. The replacement patch should be similar to the original, and maintain surface and tangent plane continuity as best as possible. Approximate patches have been proposed for Catmull-Clark surfaces [94] and the approach described in this section for Loop patches is similar.

A regular triangle B-spline can be written in Bezier form where the 15 Bezier coefficients are defined by the following points and derivatives of the surface patch (§A.1.8)[3]:

- Patch corners: $\breve{\chi}(1,0,0)$, $\breve{\chi}(0,1,0)$, and $\breve{\chi}(0,0,1)$.
- Patch edge midpoints: $\breve{\chi}\left(\frac{1}{2},\frac{1}{2},0\right)$, $\breve{\chi}\left(0,\frac{1}{2},\frac{1}{2}\right)$, and $\breve{\chi}\left(\frac{1}{2},0,\frac{1}{2}\right)$.
- Patch centre: $\breve{\chi}\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$.
- Differences of derivatives along each edge, at each patch corner: $\breve{\chi}_s(1,0,0) - \breve{\chi}_r(1,0,0)$, $\breve{\chi}_t(1,0,0) - \breve{\chi}_r(1,0,0)$, $\breve{\chi}_t(0,1,0) - \breve{\chi}_s(0,1,0)$, $\breve{\chi}_r(0,1,0) - \breve{\chi}_s(0,1,0)$, $\breve{\chi}_r(0,0,1) - \breve{\chi}_t(0,0,1)$, and $\breve{\chi}_s(0,0,1) - \breve{\chi}_t(0,0,1)$.

For a regular triangle B-spline patch the 15 recovered Bezier coefficients are linearly related to the 12 control vertices, but this restriction need not apply in general.

The Bezier form is suitable for approximating extraordinary patches because it is completely specified by geometric and derivative quantities at its boundaries and centre. Therefore, maintaining surface and tangent plane continuity at boundaries adjacent to ordinary triangle B-spline patches is straightforward (Figure A.3).

Consider an extraordinary patch with a single extraordinary vertex with valency N. All points *except* $\breve{\chi}(1,0,0)$ are evaluable after one level of subdivision using $r = 1-s-t$ and (A.7). Specifically, $\breve{\chi}(1,0,0) = \chi(\mathbf{0})$ is evaluable using the dominant left-eigenvector of the extended subdivision matrix (§A.1.5). Similarly, all derivatives *except* $\breve{\chi}_s(1,0,0) - \breve{\chi}_r(1,0,0)$ and $\breve{\chi}_t(1,0,0) - \breve{\chi}_r(1,0,0)$ are evaluable by replacing $\boldsymbol{b}(s,t)$ with $\breve{\boldsymbol{b}}(r,s,t)$ in (A.7) and differentiating (§A.1.7) [141].

For $\breve{\chi}_s(1,0,0) - \breve{\chi}_r(1,0,0)$ and $\breve{\chi}_t(1,0,0) - \breve{\chi}_r(1,0,0)$ it is *not* sensible to match the tangent vectors of the extraordinary patch because the vanishing and unbounded first derivatives of the extraordinary patch are one of the motivating reasons for constructing an approximation. Instead, following [94], $\breve{\chi}_s(1,0,0) - \breve{\chi}_r(1,0,0)$ and $\breve{\chi}_t(1,0,0) - \breve{\chi}_r(1,0,0)$ are set only to match the *direction* of the true tangent vectors, preserving the surface normal at $(1,0,0)$.

[3]The dependence of $\breve{\chi}$ and its derivatives on X has been omitted for clarity.

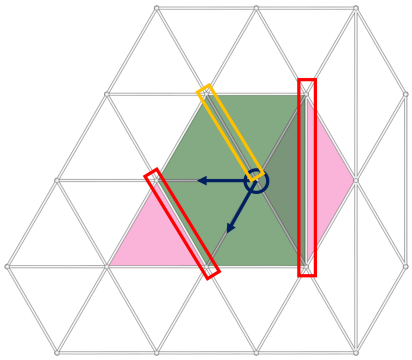


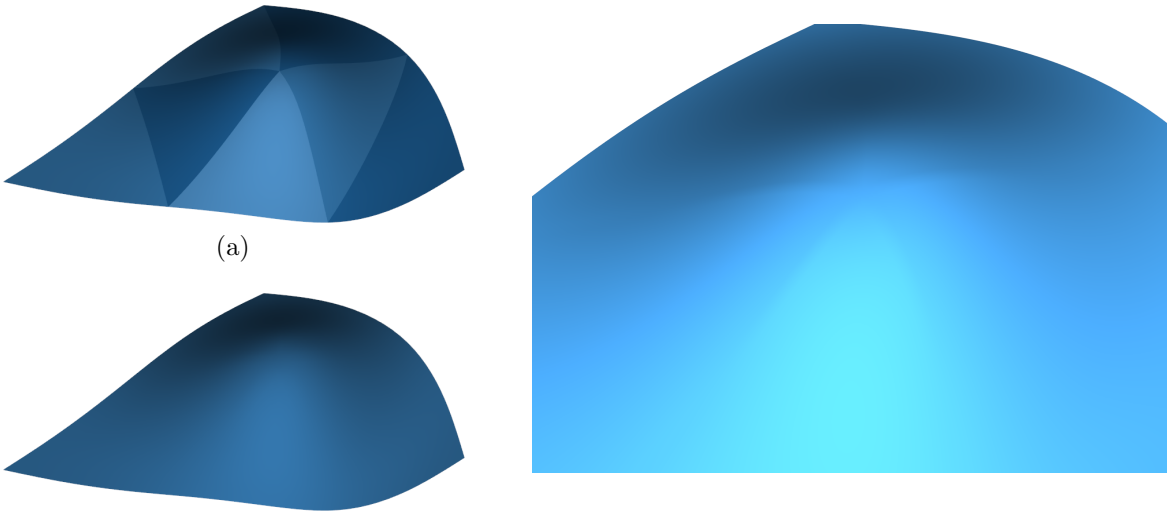
Figure A.3: An example control mesh with seven patches: two ordinary (pink) and five extraordinary (green), each with valency five. The approximate extraordinary patch should maintain geometric and tangent plane continuity with adjacent ordinary patches (red) and adjacent extraordinary patches (orange), as well as respect the geometry and direction of the tangent vectors at the extraordinary vertex (blue).

In §A.1.6 it was shown that $\tilde{\boldsymbol{v}}_2{}^\top X$ and $\tilde{\boldsymbol{v}}_3{}^\top X$ span the tangent plane at $\boldsymbol{t} = \mathbf{0}$, or $(r, s, t) = (1, 0, 0)$, where $\tilde{\boldsymbol{v}}_2$ and $\tilde{\boldsymbol{v}}_3$ are the left-eigenvectors of A corresponding to the subdominant eigenvalues. The *tangent masks*, $\tilde{\boldsymbol{v}}_s \in \mathbb{R}^{N+6}$ and $\tilde{\boldsymbol{v}}_t \in \mathbb{R}^{N+6}$, are defined as linear combinations of $\tilde{\boldsymbol{v}}_2$ and $\tilde{\boldsymbol{v}}_3$ and so that $\tilde{\boldsymbol{v}}_s{}^\top X$ and $\tilde{\boldsymbol{v}}_t{}^\top X$ point along the control mesh edges $(1, 2)$ and $(1, N+1)$:

$$\tilde{\boldsymbol{v}}_s = \begin{bmatrix} 0 & 1 & \cos(2\pi/N) & \cos(4\pi/N) & \dots & \cos(2\pi(N-1)/N) & 0 & \dots & 0 \end{bmatrix}^\top$$
$$\tilde{\boldsymbol{v}}_t = \begin{bmatrix} 0 & \cos(2\pi/N) & \cos(4\pi/N) & \dots & \cos(2\pi(N-1)/N) & 1 & 0 & \dots & 0 \end{bmatrix}^\top$$

The tangent mask scale σ is then set so that $\sigma \tilde{\boldsymbol{v}}_s{}^\top [\boldsymbol{v}_2\ \boldsymbol{v}_3] = [1\ 0]$, where $[\boldsymbol{v}_2\ \boldsymbol{v}_3]$ is the matrix of control points for the characteristic map (§A.1.6). I.e. the tangent masks are scaled so that, applied to the control vertices of the characteristic map, the tangent vectors have magnitude equal to the length of the edges $(1, 2)$ and $(1, N+1)$ (which is true for ordinary triangle B-spline patches [94]).

In summary, the approximate extraordinary patch is based on the Bezier form of the ordinary triangle B-spline and is constructed by evaluating a finite number of points and derivatives on the extraordinary patch. Vanishing and unbounded derivatives are avoided by design, and the Bezier coefficients are linearly related to the control vertices of the extraordinary patch. Unfortunately, since parametric continuity between the approximate extraordinary patch and adjacent ordinary patches (Figure A.3, red) is enforced by construction, tangent plane continuity is *not* guaranteed at the boundaries between approximate extraordinary patches (Figure A.3, orange). Potential solutions are to construct separate "tangent patches" [94], or replace the restriction of parametric



(a) (b) (c)

Figure A.4: (a) The surface defined by the control mesh in Figure A.3 with the centre five patches replaced with Bezier approximations. (b) The surface without specular highlights. (c) Tangent plane discontinuities are slightly visible as specular highlights at the joins between adjacent Bezier patches.

continuity with geometric continuity [95]. However, since the discontinuities are minor and negligible in practice (Figure A.4), these alternatives are left for future work.

A.2 Doo-Sabin Subdivision

The purpose of this section is to provide evaluation details for extraordinary Doo-Sabin patches (§A.2.1–§A.2.3) and demonstrate *why*, for extraordinary patches, second derivatives can be unbounded (§A.2.4).

A.2.1 Biquadratic B-spline Basis Functions

Assume the vertex labelling of Figure 4.5a. Let $\hat{b}_i$ denote one of three uniform quadratic B-spline basis functions given by:

$$\hat{b}_0(z) = \tfrac{1}{2}(1-z)^2, \quad \hat{b}_1(z) = -z^2 + z + \tfrac{1}{2}, \quad \hat{b}_2(z) = \tfrac{1}{2}z^2$$

where z is in the unit interval. The biquadratic B-spline basis functions are then:

$$b_i(s,t) = \hat{b}_{m_i}(s)\,\hat{b}_{n_i}(t)$$

where $\boldsymbol{m} = [1\ 1\ 0\ 0\ 0\ 1\ 2\ 2\ 2]^\top$ and $\boldsymbol{n} = [1\ 0\ 0\ 1\ 2\ 2\ 2\ 1\ 0]^\top$ specify the (zero-based index) uniform quadratic basis functions for each b_i.

A.2.2 Subdivision Matrices

Definitions for the extended subdivision matrix A, the bigger subdivision matrix $\bar{A}$, and the picking matrices P_k, $k \in \{0, 1, 2\}$ are not readily available in existing literature and are included here for completeness.

The extended subdivision matrix $A \in \mathbb{R}^{(N+5)\times(N+5)}$ is given by:

$$A = \begin{bmatrix} S & 0 \\ S_{11} & S_{12} \end{bmatrix}$$

where $S_{11} \in \mathbb{R}^{5\times N}$ and $S_{12} \in \mathbb{R}^{5\times 5}$ are given by:

$$S_{11} = \frac{1}{16}\begin{bmatrix} 3 & 0 & 0 & \cdots & 0 & 9 \\ 9 & 0 & 0 & \cdots & 0 & 3 \\ 9 & 0 & 0 & \cdots & 0 & 0 \\ 9 & 3 & 0 & \cdots & 0 & 0 \\ 3 & 9 & 0 & \cdots & 0 & 0 \end{bmatrix} \quad S_{12} = \frac{1}{16}\begin{bmatrix} 3 & 1 & 0 & 0 & 0 \\ 1 & 3 & 0 & 0 & 0 \\ 0 & 3 & 1 & 3 & 0 \\ 0 & 0 & 0 & 3 & 1 \\ 0 & 0 & 0 & 1 & 3 \end{bmatrix}$$

where columns of zeros in S_{11} are removed as necessary for $N < 5$. The $S \in \mathbb{R}^{N\times N}$ is the upper left subdivision matrix that contains the "extraordinary rules":

$$S = \begin{bmatrix} w_0 & w_1 & w_2 & \cdots & w_{N-1} \\ w_{N-1} & w_0 & w_1 & \cdots & w_{N-2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ w_1 & w_2 & w_3 & \cdots & w_0 \end{bmatrix}$$

where $\boldsymbol{w} \in \mathbb{R}^N$ is a vector of Doo-Sabin weights (zero-based index) given by [47]:

$$w_i = \begin{cases} \dfrac{N+5}{4N}, & i = 0 \\ \dfrac{3 + 2\cos\left(2\pi i/N\right)}{4N}, & 1 \leq i < N \end{cases}$$

The bigger subdivision matrix $\bar{A} \in \mathbb{R}^{(N+12)\times(N+5)}$ is:

$$\bar{A} = \begin{bmatrix} S & 0 \\ S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}$$

where $S_{21} \in \mathbb{R}^{7\times N}$ and $S_{22} \in \mathbb{R}^{7\times 5}$ are given by:

$$S_{21} = \frac{1}{16}\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 3 \\ 3 & 0 & 0 & \cdots & 0 & 1 \\ 3 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & 0 & \cdots & 0 & 0 \\ 3 & 0 & 0 & \cdots & 0 & 0 \\ 3 & 1 & 0 & \cdots & 0 & 0 \\ 1 & 3 & 0 & \cdots & 0 & 0 \end{bmatrix} \quad S_{22} = \frac{1}{16}\begin{bmatrix} 9 & 3 & 0 & 0 & 0 \\ 3 & 9 & 0 & 0 & 0 \\ 0 & 9 & 3 & 1 & 0 \\ 0 & 3 & 9 & 3 & 0 \\ 0 & 1 & 3 & 9 & 0 \\ 0 & 0 & 0 & 9 & 3 \\ 0 & 0 & 0 & 3 & 9 \end{bmatrix}$$

where columns of zeros in S_{21} are removed as necessary for $N < 5$.

By construction, the eigenvalues of S are $\lambda_1 = 1$, $\lambda_2 = \lambda_3 = 1/2$, and $\lambda_i = 1/4$ for $4 \leq i \leq N$ [47]. The additional eigenvalues for A are $\lambda_{N+1} = \lambda_{N+2} = 1/4$, $\lambda_{N+3} = \lambda_{N+4} = 1/8$, and $\lambda_{N+5} = 1/16$.

For $P_k \in \mathbb{R}^{9\times(N+12)}$, each row is filled with zeros except for a single one. The column indices of these non-zero entries for each $k \in \{0, 1, 2\}$ are:

- $k = 0$: $(N+4, N+5, 2, 1, N+2, N+3, N+10, N+11, N+12)$
- $k = 1$: $(N+3, N+4, 1, N+2, N+7, N+8, N+9, N+10, N+11)$
- $k = 2$: $(N+2, 1, N, N+1, N+6, N+7, N+8, N+3, N+4)$

A.2.3 Subdivision Functions

Let $X \in \mathbb{R}^{(N+5)\times 3}$ denote the matrix of positions, as row vectors, of the $N+5$ control vertices for an extraordinary patch with a single extraordinary face with $N \neq 4$ sides. Evaluating χ at a local patch coordinate $\boldsymbol{t} = (s, t)$ is identical to (A.7):

$$\chi(\boldsymbol{t}, X) = \boldsymbol{b}\left(t_{k,n}(\boldsymbol{t})\right)^{\top} P_k \bar{A} A^{n-1} X$$

where P_k, $\bar{A}$, and A are given in §A.2.2. For $\boldsymbol{t} = (s, t)$, the number of required subdivisions n and (zero-based) child index k are defined as:

$$n = \lfloor -\log_2(\max(s, t)) + 1 \rfloor$$
$$k = \begin{cases} 0: & 2^{-n} \leq s \leq 2^{-n+1}, 0 \leq t < 2^{-n} \\ 1: & 2^{-n} \leq s \leq 2^{-n+1}, 2^{-n} \leq t \leq 2^{-n+1} \\ 2: & 0 \leq s < 2^{-n}, 2^{-n} \leq t \leq 2^{-n+1} \end{cases}$$

with t_{k_n}:

$$t_{0,n}(\boldsymbol{t}) = (2^n s - 1, 2^n t), \quad t_{1,n}(\boldsymbol{t}) = (2^n s - 1, 2^n t - 1), \quad t_{2,n}(\boldsymbol{t}) = (2^n s, 2^n t - 1)$$

A.2.4 Finite First Derivatives and Unbounded Second Derivatives

For $N > 3$, the extended subdivision matrix A is non-diagonalisable [142]. Instead, the Jordan normal form is used to perform exact evaluation similar to (A.7). Unfortunately, the stability of first and second derivatives *cannot* be determined by simple inspection of the eigenvalues of A; the structure of the Jordan normal blocks must be considered. While a general procedure for deriving the Jordan normal form of A for arbitrary N

is beyond the scope of this section, analysis of A for $N = 3$ and $N = 6$ is presented to demonstrate the stability of first derivatives and instability of second derivatives that has been encountered in practice.

Consider a patch with an extraordinary face with $N \neq 4$ sides. Let $X \in \mathbb{R}^{(N+5)\times 3}$ denote the matrix of positions, as row vectors, of the $N + 5$ control vertices. Also, let J denote the Jordan normal form of A, and V denote the invertible matrix of generalised eigenvectors. Furthermore, let $\tilde{V}^\top = V^{-1}$. The closed-form of χ is then:

$$\begin{aligned}\chi(\boldsymbol{t}, X) &= \boldsymbol{b}(t_{k,n}(\boldsymbol{t}))^\top P_k \bar{A} A^{n-1} X \\ &= \boldsymbol{b}(t_{k,n}(\boldsymbol{t}))^\top P_k \bar{A} \left(V J^{n-1} \tilde{V}^\top\right) X \end{aligned} \tag{A.19}$$

For $N = 3$, A is diagonalisable so that $J = \Lambda$ and is given by:

$$J = \frac{1}{48}\begin{bmatrix} 48 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 24 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 24 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{8} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{16} \end{bmatrix}$$

so that (A.19) simplifies to:

$$\chi(\boldsymbol{t}, X) = \boldsymbol{b}(t_{k,n}(\boldsymbol{t}))^\top P_k \bar{A} \left(\sum_{i=1}^{N+5} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X$$

Following a similar analysis to (A.16), the first derivatives are finite since $\lambda_2 = \lambda_3 = \frac{1}{2}$:

$$\begin{aligned} &\frac{\partial}{\partial s}\left\{\chi(\boldsymbol{t}, X) - \chi(\boldsymbol{0}, X)\right\}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \\ &= \frac{\partial}{\partial s}\left\{\boldsymbol{b}(t_{k,n}(\boldsymbol{t}))^\top P_k \bar{A} \left(\sum_{i=2}^{N+5} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X\right\}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \\ &= \frac{\partial t_{0,n}(\boldsymbol{t})}{\partial s}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \boldsymbol{b}_s(t_{0,n}(\boldsymbol{\epsilon}))^\top P_0 \bar{A} \left(\sum_{i=2}^{N+5} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\ &= 2\boldsymbol{b}_s(t_{0,n}(\boldsymbol{\epsilon}))^\top P_0 \bar{A} \left(\sum_{i=2}^{N+5} (2\lambda_i)^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\ &\approx 2\boldsymbol{b}_s(t_{0,n}(\boldsymbol{\epsilon}))^\top P_0 \bar{A} \left(\boldsymbol{v}_2 \tilde{\boldsymbol{v}}_2{}^\top + \boldsymbol{v}_3 \tilde{\boldsymbol{v}}_3{}^\top\right) X \qquad \text{(A.20)} \\ &= 2\boldsymbol{b}_s(t_{0,n}(\boldsymbol{\epsilon}))^\top P_0 \bar{A} \begin{bmatrix} \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{v}}_2{}^\top \\ \tilde{\boldsymbol{v}}_3{}^\top \end{bmatrix} X \\ &= \underbrace{2\boldsymbol{b}_s(t_{0,n}(\boldsymbol{\epsilon}))^\top P_0 \bar{A} \begin{bmatrix} \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{bmatrix} T}_{\mathrm{A}} T^{-1} \begin{bmatrix} \tilde{\boldsymbol{v}}_2{}^\top \\ \tilde{\boldsymbol{v}}_3{}^\top \end{bmatrix} X \qquad \text{(A.21)} \end{aligned}$$

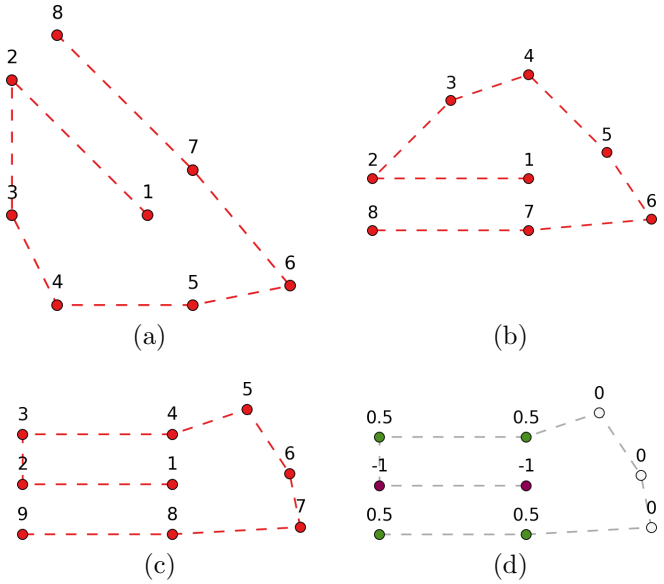


Figure A.5: (a) Control points (red) for $[\boldsymbol{v}_2\ \boldsymbol{v}_3]$. (b) Control points for $[\boldsymbol{v}_2\ \boldsymbol{v}_3]\,T$. Edges $(1,2)$ and $(7,8)$ are parallel. (c) Control points for $P_0\bar{A}\,[\boldsymbol{v}_2\ \boldsymbol{v}_3]\,T$. Edges $(1,2)$, $(3,4)$ and $(8,9)$ are parallel and equidistant. (d) Weights (purple to green) for $\boldsymbol{b}_{ss}\,(\boldsymbol{t})$ where $\boldsymbol{t} = (s, 0)$.

where $k = 0$ because $\boldsymbol{\epsilon} = (\epsilon, 0)$, and (A.20) follows since $\lambda_2 = \lambda_3 > \lambda_4 \geq \cdots \geq \lambda_8$. In (A.21), $T \in \mathbb{R}^{2\times 2}$ is an invertible matrix so that the edges $(1,2)$ and $(7,8)$ of $[\boldsymbol{v}_2\ \boldsymbol{v}_3]\,T$ are parallel at a distance of one (Figure A.5). In this case, A evaluates to $[0\ -1]$ for all $\boldsymbol{t} = (s, 0)$. Therefore:

$$\begin{aligned}
\frac{\partial}{\partial s}\left\{\chi\left(\boldsymbol{t}, X\right) - \chi\left(\boldsymbol{0}, X\right)\right\}\Big|_{\boldsymbol{t}=\boldsymbol{\epsilon}} &\approx \begin{bmatrix}0 & -1\end{bmatrix} T^{-1} \begin{bmatrix}\tilde{\boldsymbol{v}}_2{}^\top \\ \tilde{\boldsymbol{v}}_3{}^\top\end{bmatrix} X \\
&= \begin{bmatrix}0 & -1\end{bmatrix} \begin{bmatrix}\frac{1}{3} & \frac{1}{3} & 0 & 0 & \cdots & 0 \\ -\frac{1}{3} & -\frac{1}{3} & \frac{2}{3} & 0 & \cdots & 0\end{bmatrix} X \\
&= \frac{1}{3}\boldsymbol{x}_1 + \frac{1}{3}\boldsymbol{x}_2 - \frac{2}{3}\boldsymbol{x}_3
\end{aligned}$$

which has been confirmed by experiments. A similar result can be demonstrated for χ_t.

Now consider evaluating χ_{ss} near $\boldsymbol{t} = (s,t) = \mathbf{0}$:

$$\begin{aligned}
&\chi_{ss}\left(\boldsymbol{t}, X\right)|_{\boldsymbol{t}=\boldsymbol{\epsilon}} \\
&= \left(\left.\frac{\partial t_{0,n}\left(\boldsymbol{t}\right)}{\partial s}\right|_{\boldsymbol{t}=\boldsymbol{\epsilon}}\right)^2 \boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0 \bar{A}\left(\sum_{i=2}^{N+5} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\
&= 4^n \boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0 \bar{A}\left(\lambda_2^{n-1}\begin{bmatrix}\boldsymbol{v}_2 & \boldsymbol{v}_3\end{bmatrix} T T^{-1}\begin{bmatrix}\tilde{\boldsymbol{v}}_2{}^\top \\ \tilde{\boldsymbol{v}}_3{}^\top\end{bmatrix} + \sum_{i=4}^{N+5} \lambda_i^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \\
&= 4 \boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0 \bar{A}\left(\sum_{i=4}^{N+5} \left(4\lambda_i\right)^{n-1} \boldsymbol{v}_i \tilde{\boldsymbol{v}}_i{}^\top\right) X \qquad \text{(A.22)} \\
&\approx 4\left(4\lambda_4\right)^{n-1} \boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0 \bar{A}\left(\boldsymbol{v}_4 \tilde{\boldsymbol{v}}_4{}^\top + \boldsymbol{v}_5 \tilde{\boldsymbol{v}}_5{}^\top\right) X
\end{aligned}$$

which is finite since $\lambda_4 = \lambda_5 = \frac{1}{4}$. (A.22) follows because $P_0 \bar{A}\left[\boldsymbol{v}_2\ \boldsymbol{v}_3\right] T$ has parallel edges $(1,2)$, $(3,4)$, $(8,9)$ by construction (Figure A.5c), and the subsequent inner product with $\boldsymbol{b}_{ss}\left(\boldsymbol{t}\right)$ (Figure A.5d) is $\mathbf{0}$.

Now consider $N = 6$ for which the Jordan normal form is:

$$J = \frac{1}{48}\begin{bmatrix}
48 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 24 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 12 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 12 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 12 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 6 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3
\end{bmatrix}$$

The Jordan blocks in J are for the eigenvalues $\lambda_4 = \lambda_5 = \ldots = \lambda_7 = \frac{1}{4}$. Therefore, demonstrating that first derivatives are finite — which depends on λ_2 and λ_3 — is straightforward and similar to the analysis for $N = 3$.

For the evaluation of second derivatives, the first three eigenvalues also have no impact. Using the formula for repeated multiplication of a Jordan block, χ_{ss} at $\boldsymbol{t} = \boldsymbol{\epsilon}$ is:

$$\begin{aligned}
\chi_{ss}\left(\boldsymbol{\epsilon}, X\right) \approx 4^n \boldsymbol{b}_{ss}\left(t_{0,n}\left(\boldsymbol{\epsilon}\right)\right)^\top P_0 \bar{A}\Bigg(&\begin{bmatrix}\boldsymbol{v}_4 & \boldsymbol{v}_5\end{bmatrix}\begin{bmatrix}\lambda_4^{n-1} & \left(\frac{n-1}{48}\right)\lambda_4^{n-2} \\ 0 & \lambda_4^{n-1}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{v}}_4{}^\top \\ \tilde{\boldsymbol{v}}_5{}^\top\end{bmatrix} + \\
&\begin{bmatrix}\boldsymbol{v}_6 & \boldsymbol{v}_7\end{bmatrix}\begin{bmatrix}\lambda_4^{n-1} & \left(\frac{n-1}{48}\right)\lambda_4^{n-2} \\ 0 & \lambda_4^{n-1}\end{bmatrix}\begin{bmatrix}\tilde{\boldsymbol{v}}_6{}^\top \\ \tilde{\boldsymbol{v}}_7{}^\top\end{bmatrix} + \\
&\lambda_4^{n-1} \boldsymbol{v}_8 \tilde{\boldsymbol{v}}_8{}^\top\Bigg) X
\end{aligned}$$

and using $\lambda_4 = \frac{1}{4}$:

$$\chi_{ss}(\boldsymbol{\epsilon}, X) \approx 4\boldsymbol{b}_{ss}(t_{0,n}(\boldsymbol{\epsilon}))^\top P_0 \bar{A} \Bigg(\begin{bmatrix} \boldsymbol{v}_4 & \boldsymbol{v}_5 \end{bmatrix} \begin{bmatrix} 1 & \left(\frac{n-1}{12}\right) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{v}}_4{}^\top \\ \tilde{\boldsymbol{v}}_5{}^\top \end{bmatrix} + \begin{bmatrix} \boldsymbol{v}_6 & \boldsymbol{v}_7 \end{bmatrix} \begin{bmatrix} 1 & \left(\frac{n-1}{12}\right) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tilde{\boldsymbol{v}}_6{}^\top \\ \tilde{\boldsymbol{v}}_7{}^\top \end{bmatrix} + \boldsymbol{v}_8 \tilde{\boldsymbol{v}}_8{}^\top \Bigg) X$$

which is unbounded due to the $\frac{n-1}{12}$ terms. This has been verified in experiments.

References

[1] E. L. Abel. *Fetal alcohol syndrome*. Springer, 1984.

[2] S. Agarwal, K. Mierle, et al. Ceres solver. `http://www.ceres-solver.org`, 2010–.

[3] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE TSP*, 54(11):4311–4322, 2006.

[4] B. Amberg and T. Vetter. GraphTrack: Fast and globally optimal tracking in videos. In *IEEE CVPR*, pages 1209–1216, 2011.

[5] P. R. Amestoy, T. A. Davis, and I. S. Duff. An approximate minimum degree ordering algorithm. *SIAM Journal on Matrix Analysis and Applications*, 17(4):886–905, 1996.

[6] B. Appleton and C. Sun. Circular shortest paths by branch and bound. *Pattern Recognition*, 36(11):2513–2520, 2003.

[7] W. F. Armstrong and T. Ryan. *Feigenbaum's echocardiography*. Lippincott Williams & Wilkins, 2012.

[8] N. Aspert, D. S. Cruz, and T. Ebrahimi. MESH: Measuring errors between surfaces using the Hausdorff distance. In *IEEE ICME*, pages 705–708, 2002.

[9] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with applications to tracking and navigation: Theory algorithms and software*. Wiley, 2004.

[10] D. Barbosa, T. Dietenbeck, J. Schaerer, J. D'hooge, D. Friboulet, and O. Bernard. B-spline explicit active surfaces: An efficient framework for real-time 3-D region-based segmentation. *IEEE TIP*, 21(1):241–251, 2012.

[11] D. Barbosa, T. Dietenbeck, B. Heyde, H. Houle, D. Friboulet, J. Dhooge, and O. Bernard. Fast and fully automatic 3-D echocardiographic segmentation using B-spline explicit active surfaces: Feasibility study and validation in a clinical setting. *Ultrasound in medicine & biology*, 39(1):89–101, 2013.

[12] G. Behiels, F. Maes, D. Vandermeulen, and P. Suetens. Evaluation of image features and search strategies for segmentation of bone structures in radiographs using active shape models. *Medical Image Analysis*, 6(1):47–62, 2002.

[13] A. Belaid, D. Boukerroui, Y. Maingourd, and J. Lerallut. Implicit active contours for ultrasound images segmentation driven by phase information and local maximum likelihood. In *IEEE ISBI*, pages 630–635, 2011.

[14] O. Bernard, D. Friboulet, P. Thévenaz, and M. Unser. Variational B-spline level-set: A linear filtering approach for fast deformable model evolution. *IEEE TIP*, 18(6): 1179–1191, 2009.

[15] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE TPAMI*, 14(2):239–256, 1992.

[16] C. M. Bishop et al. *Pattern recognition and machine learning*, volume 1. Springer, 2006.

[17] A. Björck. *Numerical methods for least squares problems*. SIAM, 1996.

[18] A. Blake, M. Isard, et al. *Active contours*, volume 1. Springer, 2000.

[19] L. N. Bohs and G. E. Trahey. A novel method for angle independent ultrasonic imaging of blood flow and tissue motion. *IEEE TBME*, 38(3):280–286, 1991.

[20] I. Boier-Martin and D. Zorin. Differentiable parameterization of catmull-clark subdivision surfaces. In *Symposium on Geometry Processing*, pages 155–164. ACM, 2004.

[21] C. F. Borges and T. Pastva. Total least squares fitting of Bézier and B-spline curves to ordered data. *Computer Aided Geometric Design*, 19(4):275–289, 2002.

[22] J. G. Bosch, S. C. Mitchell, B. P. Lelieveldt, F. Nijland, O. Kamp, M. Sonka, and J. H. Reiber. Automatic segmentation of echocardiographic sequences by active appearance motion models. *IEEE TMI*, 21(11):1374–1383, 2002.

[23] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE TPAMI*, 23(11):1222–1239, 2001.

[24] A. D. Brett and C. J. Taylor. A method of automated landmark generation for automated 3D PDM construction. *Image and Vision Computing*, 18(9):739–748, 2000.

[25] T. Brox and D. Cremers. On the statistical interpretation of the piecewise smooth mumford-shah functional. In *Scale Space and Variational Methods in Computer Vision*, pages 203–213. Springer, 2007.

[26] J. Canny. A computational approach to edge detection. *IEEE TPAMI*, (6):679–698, 1986.

[27] V. Caselles, R. Kimmel, and G. Sapiro. Geodesic active contours. *IJCV*, 22(1):61–79, 1997.

[28] T. J. Cashman and A. W. Fitzgibbon. What shape are dolphins? Building 3D morphable models from 2D images. *IEEE TPAMI*, 35(1):232–244, 2013.

[29] E. Catmull and J. Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[30] J. Caudron, J. Fares, V. Lefebvre, P. Vivier, C. Petitjean, and J. Dacher. Cardiac MRI assessment of right ventricular function in acquired heart disease: Factors of variability. *Academic Radiology*, 19(8):991–1002, 2012.

[31] T. F. Chan and L. A. Vese. Active contours without edges. *IEEE TIP*, 10(2):266–277, 2001.

[32] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145–155, 1992.

[33] Y. Chen, T. A. Davis, W. W. Hager, and S. Rajamanickam. Algorithm 887: CHOLMOD, supernodal sparse Cholesky factorization and update/downdate. *ACM TOMS*, 35(3):22, 2008.

[34] K. Cheng, W. Wang, H. Qin, K. Wong, H. Yang, and Y. Liu. Design and analysis of optimization methods for subdivision surface fitting. *IEEE TVCG*, 13(5):878–890, 2007.

[35] Y. Cheng. Mean shift, mode seeking, and clustering. *IEEE TPAMI*, 17(8):790–799, 1995.

[36] T. F. Cootes, C. J. Taylor, D. H. Cooper, and J. Graham. Active shape models — their training and application. *Computer Vision and Image Understanding*, 61(1): 38–59, 1995.

[37] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. In *ECCV*, pages 484–498. Springer, 1998.

[38] T. F. Cootes, G. J. Edwards, and C. J. Taylor. Active appearance models. *IEEE TPAMI*, 23(6):681–685, 2001.

[39] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[40] A. Criminisi, J. Shotton, and S. Bucciarelli. Decision forests with long-range spatial context for organ localization in CT volumes. *MICCAI PMMIA*, pages 69–80, 2009.

[41] C. Davatzikos, X. Tao, and D. Shen. Hierarchical active shape models, using the wavelet transform. *IEEE TMI*, 22(3):414–423, 2003.

[42] R. H. Davies, C. J. Twining, T. F. Cootes, J. C. Waterton, and C. J. Taylor. 3D statistical shape models using direct optimisation of description length. In *ECCV*, pages 3–20. Springer, 2002.

[43] J. D'hooge, B. Bijnens, J. Thoen, F. V. d. Werf, G. R. Sutherland, and P. Suetens. Echocardiographic strain and strain-rate imaging: A new tool to study regional myocardial function. *IEEE TMI*, 21(9):1022–1030, 2002.

[44] P. Dollár and C. L. Zitnick. Structured forests for fast edge detection. In *IEEE ICCV*, 2013.

[45] J. S. Domingos, R. V. Stebbing, P. Leeson, and J. A. Noble. Structured random forests for myocardium delineation in 3D echocardiography. In *MICCAI Workshop on Machine Learning in Medical Imaging*, 2014.

[46] J. S. Domingos, R. V. Stebbing, and J. A. Noble. Endocardial segmentation using structured random forests in 3D echocardiography. In *MICCAI Challenge on Endocardial Three-dimensional Ultrasound Segmentation*, 2014.

[47] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[48] A. Doucet, N. De Freitas, and N. Gordon. *Sequential Monte Carlo methods in practice*. Springer, 2001.

[49] Q. Duan, E. D. Angelini, and A. F. Laine. Surface function actives. *Journal of Visual Communication and Image Representation*, 20(7):478–490, 2009.

[50] Q. Duan, E. D. Angelini, and A. F. Laine. Real-time segmentation by active geometric functions. *Computer Methods and Programs in Biomedicine*, 98(3):223–230, 2010.

[51] S. Essafi, G. Langs, and N. Paragios. Hierarchical 3D diffusion wavelet shape priors. In *IEEE CVPR*, pages 1717–1724, 2009.

[52] P. Felzenszwalb and D. Huttenlocher. Distance transforms of sampled functions. Technical report, Cornell University, 2004.

[53] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *IJCV*, 61(1):55–79, 2005.

[54] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. *Image and Vision Computing*, 21(13):1145–1153, 2003.

[55] Y. Freund and R. E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory*, pages 23–37. Springer, 1995.

[56] J. Friedman, T. Hastie, R. Tibshirani, et al. Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

[57] B. H. Friemel, L. N. Bohs, and G. E. Trahey. Relative performance of two-dimensional speckle-tracking techniques: Normalized correlation, non-normalized correlation and sum-absolute-difference. In *IEEE Ultrasonics Symposium*, volume 2, pages 1481–1484, 1995.

[58] P. Geurts, D. Ernst, and L. Wehenkel. Extremely randomized trees. *Machine Learning*, 63(1):3–42, 2006.

[59] G. Golub and V. Pereyra. Separable nonlinear least squares: The variable projection method and its applications. *Inverse problems*, 19(2):R1, 2003.
[60] H. Gouraud. Continuous shading of curved surfaces. *IEEE TC*, 100(6):623–629, 1971.
[61] J. Guerrero, S. E. Salcudean, J. A. McEwen, B. A. Masri, and S. Nicolaou. Real-time vessel segmentation and tracking for ultrasound imaging applications. *IEEE TMI*, 26 (8):1079–1090, 2007.
[62] J. E. Hall. *Guyton and Hall Textbook of Medical Physiology: Enhanced E-book*. Elsevier, 2010.
[63] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. In *SIGGRAPH*, pages 35–44. ACM, 1993.
[64] P. L. Hammer, P. Hansen, and B. Simeone. Roof duality, complementation and persistency in quadratic 0–1 optimization. *Mathematical Programming*, 28(2):121–155, 1984.
[65] B. Hearn-Stebbins. Normal fetal growth assessment: A review of literature and current practice. *Journal of Diagnostic Medical Sonography*, 11(4):176–187, 1995.
[66] H. Helfrich and D. Zwick. A trust region algorithm for parametric curve and surface fitting. *Journal of Computational and Applied Mathematics*, 73(1):119–134, 1996.
[67] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH*, pages 295–302. ACM, 1994.
[68] J. Hoschek, D. Lasser, and L. L. Schumaker. *Fundamentals of computer aided geometric design*. AK Peters, Ltd., 1993.
[69] P. R. Hoskins, K. Martin, and A. Thrush. *Diagnostic ultrasound: Physics and equipment*. Cambridge University Press, 2010.
[70] X. Huang, D. P. Dione, C. B. Compas, X. Papademetris, B. A. Lin, A. J. Sinusas, and J. S. Duncan. A dynamical appearance model based on multiscale sparse representation: Segmentation of the left ventricle from 4D echocardiography. In *MICCAI*, pages 58–65. Springer, 2012.
[71] X. Huang, B. A. Lin, C. B. Compas, A. J. Sinusas, L. H. Staib, and J. S. Duncan. Segmentation of left ventricles from echocardiographic sequences via sparse appearance representation. In *IEEE MMBIA Workshop*, pages 305–312, Jan 2012.
[72] X. Huang, D. P. Dione, B. A. Lin, A. Bregasi, A. J. Sinusas, and J. S. Duncan. Segmentation of 4d echocardiography using stochastic online dictionary learning. In *MICCAI*, pages 57–65. Springer, 2013.
[73] P. J. Huber. Robust estimation of a location parameter. *The Annals of Mathematical Statistics*, 35(1):73–101, 1964.
[74] M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *ECCV*, pages 343–356. Springer, 1996.
[75] M. Isard and A. Blake. Condensation — conditional density propagation for visual tracking. *IJCV*, 29(1):5–28, 1998.
[76] W. Jeong and C. H. Kim. Direct reconstruction of displaced subdivision surface from unorganized points. In *IEEE CGA*, pages 160–168, 2001.
[77] H. F. Kaiser. The varimax criterion for analytic rotation in factor analysis. *Psychometrika*, 23(3):187–200, 1958.
[78] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active contour models. *IJCV*, 1(4): 321–331, 1988.
[79] L. Kaufman and P. J. Rousseeuw. *Finding groups in data: An introduction to cluster*

analysis, volume 344. Wiley, 2009.

[80] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts — A review. *IEEE TPAMI*, 29(7):1274–1279, 2007.

[81] P. Kovesi. *Invariant measures of image features from phase information.* PhD thesis, 1996.

[82] D. Kucera and R. W. Martin. Segmentation of sequences of echocardiographic images using a simplified 3d active contour model with region-based external forces. *Computerized Medical Imaging and Graphics*, 21(1):1–21, 1997.

[83] S. Lankton. Sparse field methods — technical report. *Georgia Institute of Technology*, 2009.

[84] S. Lankton and A. Tannenbaum. Localizing region-based active contours. *IEEE TIP*, 17(11):2029–2039, 2008.

[85] A. Lee, H. Moreton, and H. Hoppe. Displaced subdivision surfaces. In *SIGGRAPH*, pages 85–94. ACM, 2000.

[86] P. Leeson, D. Augustine, A. Mitchell, and H. Becher. *Oxford Specialist Handbooks in Cardiology: Echocardiography.* Oxford University Press, 2012.

[87] V. Lempitsky, M. Verhoek, J. A. Noble, and A. Blake. Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography. In *Functional Imaging and Modeling of the Heart*, pages 447–456. Springer, 2009.

[88] K. E. Leung and J. G. Bosch. Local wall-motion classification in echocardiograms using shape models and orthomax rotations. In *Functional Imaging and Modeling of the Heart*, pages 1–11. Springer, 2007.

[89] K. Levenberg. A method for the solution of certain nonlinear problems in least squares. *Quaterley of Applied Mathematics*, 2:164–168, 1944.

[90] Y. Li, T. Tan, I. Volkau, and W. L. Nowinski. Model-guided segmentation of 3D neuroradiological image using statistical surface wavelet model. In *IEEE CVPR*, pages 1–7, 2007.

[91] N. Litke, A. Levin, and P. Schröder. Fitting subdivision surfaces. In *IEEE VIS*, pages 319–324, 2001.

[92] Y. Liu and W. Wang. A revisit to least squares orthogonal distance fitting of parametric curves and surfaces. In *Advances in Geometric Modeling and Processing*, pages 384–397. Springer, 2008.

[93] C. Loop. Smooth subdivision surfaces based on triangles. 1987.

[94] C. Loop and S. Schaefer. Approximating Catmull-Clark subdivision surfaces with bicubic patches. *ACM TOG*, 27(1):8, 2008.

[95] C. Loop and S. Schaefer. G2 tensor product splines over extraordinary vertices. In *Computer Graphics Forum*, volume 27, pages 1373–1382. Wiley, 2008.

[96] K. Madsen, H. Bruun, and O. Tingleff. Methods for non-linear least squares problems. 1999.

[97] M. Marinov and L. Kobbelt. Optimization techniques for approximation with subdivision surfaces. In *Symposium on Solid Modeling and Applications*, pages 113–122. ACM, 2004.

[98] M. Marinov and L. Kobbelt. Optimization methods for scattered data approximation with subdivision surfaces. *Graphical Models*, 67(5):452–473, 2005.

[99] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial & Applied Mathematics*, 11(2):431–441, 1963.

[100] T. McInerney and D. Terzopoulos. Deformable models in medical image analysis: A

survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[101] S. K. Mitra and Y. Kuo. *Digital signal processing: A computer-based approach*, volume 2. McGraw-Hill, 2006.

[102] J. Montagnat, H. Delingette, and N. Ayache. A review of deformable surfaces: Topology, geometry and deformation. *Image and Vision Computing*, 19(14):1023–1040, 2001.

[103] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi. Entangled decision forests and their application for semantic segmentation of CT images. In *IPMI*, pages 184–196. Springer, 2011.

[104] J. H. Morra, Z. Tu, L. G. Apostolova, A. E. Green, A. W. Toga, and P. M. Thompson. Comparison of AdaBoost and support vector machines for detecting Alzheimer's disease through automated hippocampal segmentation. *IEEE TMI*, 29(1):30–43, 2010.

[105] S. S. Muchnick. *Advanced compiler design implementation*. Morgan Kaufmann, 1997.

[106] D. Mumford and J. Shah. Boundary detection by minimizing functionals. *Image Understanding*, pages 19–43, 1988.

[107] D. Nain, S. Haker, A. Bobick, and A. R. Tannenbaum. Multiscale 3D shape analysis using spherical wavelets. In *MICCAI*, pages 459–467. Springer, 2005.

[108] D. Nain, S. Haker, A. Bobick, and A. Tannenbaum. Shape-driven 3D segmentation using spherical wavelets. In *MICCAI*, pages 66–74. Springer, 2006.

[109] A. I. L. Namburete, R. V. Stebbing, and J. A. Noble. Cranial parametrization of the fetal head for 3D ultrasound image analysis. In *MIUA*, pages 196–201, 2013.

[110] A. I. L. Namburete, R. V. Stebbing, and J. A. Noble. Diagnostic plane extraction from 3D parametric surface of the fetal cranium. In *MIUA*, pages 27–32, 2014.

[111] A. I. L. Namburete, M. Yaqub, B. Kemp, A. T. Papageorghiou, and J. A. Noble. Predicting fetal neurodevelomental age from ultrasound images. In *MICCAI*, pages 260–267. Springer, 2014.

[112] T. R. Nelson and D. H. Pretorius. Three-dimensional ultrasound imaging. *Ultrasound in Medicine & Biology*, 24(9):1243–1270, 1998.

[113] N. S. Nise. *Control systems engineering*. Wiley, 2007.

[114] J. Nocedal and S. J. Wright. Numerical Optimization 2nd. 2006.

[115] A. Opelt, A. Pinz, and A. Zisserman. A boundary-fragment-model for object detection. In *ECCV*, pages 575–588. Springer, 2006.

[116] F. Orderud. A framework for real-time left ventricular tracking in 3D+T echocardiography, using nonlinear deformable contours and Kalman filter based tracking. In *IEEE Computers in Cardiology*, pages 125–128, 2006.

[117] F. Orderud, J. Hansgård, and S. I. Rabben. Real-time tracking of the left ventricle in 3D echocardiography using a state estimation approach. In *MICCAI*, pages 858–865. Springer, 2007.

[118] F. Orderud, G. Kiss, S. Langeland, E. W. Remme, H. Torp, and S. I. Rabben. Real-time left ventricular speckle-tracking in 3D echocardiography with deformable subdivision surfaces. In *MICCAI Workshop on Analysis of Functional Medical Images*, pages 41–48, 2008.

[119] S. Osher and J. A. Sethian. Fronts propagating with curvature-dependent speed: Algorithms based on Hamilton-Jacobi formulations. *Journal of Computational Physics*, 79(1):12–49, 1988.

[120] P. Perez, J. Vermaak, and A. Blake. Data fusion for visual tracking with particles. *Proceedings of the IEEE*, 92(3):495–513, 2004.

[121] D. Pescia, N. Paragios, and S. Chemouny. Automatic detection of liver tumors. In *IEEE ISBI*, pages 672–675, 2008.

[122] C. Petitjean and J. Dacher. A review of segmentation methods in short axis cardiac MR images. *Medical Image Analysis*, 15(2):169–184, 2011.

[123] B. T. Phong. Illumination for computer generated pictures. *Communications of the ACM*, 18(6):311–317, 1975.

[124] L. Piegl and W. Tiller. *Curve and Surface Basics*. Springer, 1995.

[125] E. Pierce. Diagram of the human heart. `http://commons.wikimedia.org/wiki/File:Diagram_of_the_human_heart.svg`.

[126] H. Pottmann and M. Hofer. *Geometry of the squared distance function to curves and surfaces*. Springer, 2003.

[127] H. Pottmann and S. Leopoldseder. A concept for parametric surface fitting which avoids the parametrization problem. *Computer Aided Geometric Design*, 20(6):343–362, 2003.

[128] H. Pottmann, S. Leopoldseder, and M. Hofer. Approximation with active B-spline curves and surfaces. In *IEEE CGA*, pages 8–25, 2002.

[129] T. Rackham. *Ultrasound segmentation tools and their application to assess fetal nutritional health*. PhD thesis, 2014.

[130] K. Rajpoot, V. Grau, and J. A. Noble. Local-phase based 3D boundary detection using monogenic signal and its application to real-time 3-D echocardiography images. In *IEEE ISBI*, pages 783–786, 2009.

[131] C. R. Rao and S. K. Mitra. *Generalized inverse of matrices and its applications*, volume 7. Wiley, 1971.

[132] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

[133] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, pages 416–431, 1983.

[134] S. Rueda, S. Fathima, C. L. Knight, M. Yaqub, A. T. Papageorghiou, B. Rahmatullah, A. Foi, M. Maggioni, A. Pepe, J. Tohka, R. V. Stebbing, J. E. McManigle, A. Ciurte, X. Bresson, M. B. Cuadra, C. Sun, G. V. Ponomarev, M. S. Gelfand, M. D. Kazanov, C. Wang, H. Chen, C. Peng, C. Hung, and J. A. Noble. Evaluation and comparison of current fetal ultrasound image segmentation methods for biometric measurements: A grand challenge. *IEEE TMI*, 2014.

[135] A. Sarti, C. Corsi, E. Mazzini, and C. Lamberti. Maximum likelihood segmentation of ultrasound images with rayleigh distribution. *IEEE TUFFC*, 52(6):947–960, 2005.

[136] J. A. Sethian. *Level set methods and fast marching methods: Evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, volume 3. Cambridge University Press, 1999.

[137] Y. Shi and W. C. Karl. A real-time algorithm for the approximation of level-set-based curve evolution. *IEEE TIP*, 17(5):645–656, 2008.

[138] J. Shotton, A. Blake, and R. Cipolla. Multiscale categorical object recognition using contour fragments. *IEEE TPAMI*, 30:1270–1281, July 2008.

[139] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *IEEE CVPR*, pages 1–8, 2008.

[140] T. Speer, M. Kuppe, and J. Hoschek. Global reparametrization for curve approximation. *Computer Aided Geometric Design*, 15(9):869–877, 1998.

[141] J. Stam. Evaluation of Loop subdivision surfaces. 1998.

[142] J. Stam. Exact evaluation of catmull-clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH*, pages 395–404. ACM, 1998.

[143] R. V. Stebbing and J. E. McManigle. A boundary fragment model for head segmentation in fetal ultrasound. *IEEE ISBI Proceedings of Challenge US*, pages 9–11, 2012.

[144] R. V. Stebbing and J. A. Noble. Delineating anatomical boundaries using the boundary fragment model. *Medical Image Analysis*, 17(8):1123–1136, 2013.

[145] R. V. Stebbing, J. E. McManigle, and J. A. Noble. Interpreting edge information for improved endocardium delineation in echocardiograms. In *IEEE ISBI*, pages 238–241, 2012.

[146] M. B. Stegmann, K. Sjöstrand, and R. Larsen. Sparse modeling of landmark and texture variability using the orthomax criterion. In *Medical Imaging*. International Society for Optics and Photonics, 2006.

[147] C. Studholme. Mapping fetal brain development in utero using magnetic resonance imaging: The Big Bang of brain mapping. *Annual Review of Biomedical Engineering*, 13(1):345–368, 2011.

[148] H. Suzuki, S. Takeuchi, and T. Kanai. Subdivision surface fitting to a range of points. In *IEEE CGA*, pages 158–167, 1999.

[149] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *IEEE CVPR*, pages 103–110, 2012.

[150] J. Taylor, R. V. Stebbing, V. Ramakrishna, C. Keskin, J. Shotton, S. Izadi, A. Hertzmann, and A. W. Fitzgibbon. User-specific hand modeling from monocular depth sequences. In *IEEE CVPR*, 2014.

[151] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – A modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372. Springer, 2000.

[152] Z. Tu and X. Bai. Auto-context and its application to high-level vision tasks and 3D brain image segmentation. *IEEE TPAMI*, 32(10):1744–1757, 2010.

[153] W. Wang, H. Pottmann, and Y. Liu. Fitting B-spline curves to point clouds by curvature-based squared distance minimization. *ACM TOG*, 25(2):214–238, 2006.

[154] Y. Wang, B. S. Peterson, and L. H. Staib. Shape-based 3D surface correspondence using geodesics and local geometry. In *IEEE CVPR*, volume 2, pages 644–651, 2000.

[155] R. T. Whitaker. A level-set approach to 3D reconstruction from range data. *IJCV*, 29 (3):203–231, 1998.

[156] F. Yeung, S. F. Levinson, and K. J. Parker. Multilevel and motion model-based ultrasonic speckle tracking algorithms. *Ultrasound in Medicine & Biology*, 24(3): 427–441, 1998.

[157] W. Yu, N. Lin, P. Yan, K. Purushothaman, A. Sinusas, K. Thiele, and J. S. Duncan. Motion analysis of 3D ultrasound texture patterns. In *Functional Imaging and Modeling of the Heart*, pages 253–261. Springer, 2003.

[158] X. Zhou, X. Huang, J. S. Duncan, and W. Yu. Active contours with group similarity. In *IEEE CVPR*, pages 2969–2976, 2013.