NTNU
Innovation and Creativity

Automatic Real-time Left Ventricular Tracking in 3D Echocardiography

Using probabilistic contour tracking and sequential state estimation algorithms

Fredrik Orderud

Master of Science in Computer Science
Submission date: June 2006
Supervisor: Bjørn Olstad, IDI
Co-supervisor: Sigmund Frigstad, GE Vingmed Ultrasound AS

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

The purpose of this project is to develop a framework for automatic real-time tracking of the dominant left ventricular motion throughout the heart cycle in 3D ultrasound.

The tracking technique explored is probabilistic contour tracking, based on deformable contour models of the left ventricle. The problem is formulated as a parameter estimation problem, and the tracking is to be performed by means of sequential state estimation algorithms.

Assignment given: 09. January 2006
Supervisor: Bjørn Olstad, IDI

Preface

This is a master thesis for the Master of Technology program at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU). The assignment was given by GE Vingmed Ultrasound AS (GEVU). The report and underlying work has been done in five months, in the spring of 2006.

As a student attending the integrated PhD program at IDI this thesis also contains a research plan for the remaining part of my PhD. The research plan is provided as an appendix, and outlines a plan for future work that will be greatly based on the results presented in this thesis.

Acknowledgment

I would like to thank Sigmund Frigstad at GEVU for useful discussions and suggestions on potential applications for the tracking technology within the field of echocardiography. I also want to thank my main supervisor Bjørn Olstad at IDI/NTNU for constructive feedback throughout the writing of this thesis.

In addition, I want to thank GEVU for access to the 81 patient dataset used for testing and tuning of the framework, and Brage Amundsen at ISB/NTNU for the 21 patient dataset used for independent validation of the method.

regards
Fredrik Orderud

Abstract

This thesis presents a new framework for automatic real-time left ventricular (LV) tracking in 3D+T echocardiography. The framework enables usage of existing biomechanical deformation models for the heart, with nonlinear modes of deformation, combined with edge models for the endocardium boundary.

Tracking is performed in a sequential state estimation fashion, using an extended Kalman filter to recursively predict and update contour deformations in real-time. Contours are detected using normal-displacement measurements from points on the predicted contour, and are processed efficiently using an information-filter formulation of the Kalman filter.

Promising results are shown for LV-tracking using a truncated ellipsoid contour model, with deformation parameters for translation, rotation, scaling and bending in all three dimensions. The tracking framework automatically detects LV position initially, even in situations where it is partially outside the volume or the initial contour is inaccurately placed. It also successfully tracks the dominant motion throughout the heart cycle, and correctly identifies the long-axis.

Primary scientific contributions:

- Extension of existing contour tracking framework (Blake et al.) to tracking in 3D-data.
- Generalization of existing contour tracking framework (Blake et al.) to all types of contours, deformed by arbitrary, nonlinear deformations.
- Derivation of, and extensions to the information filter/sequential estimation equations by Blake et al. to allow more efficient processing of independent measurements.
- Experimental validation of feasibility of real-time left ventricular tracking in 3D echocardiography.

Contents

Part I

Introduction

Chapter 1

Background & Motivation

The main goal for this thesis it to develop a framework capable of automatic real-time left ventricular (LV) tracking in 3D+T echocardiography, using video tracking algorithms. This goal is motivated by a more long-term vision of automatic cardiac diagnosis, based on ultrasound recordings. Successful tracking of high quality will hopefully enable automatic extraction of clinical parameters from the recordings, which can then be used to detect cardiac diseases.

Video tracking is a broad and diverse field of research, encompassing many different techniques and methodologies. The objective is usually the discovery and following of objects in video streams. Most research can roughly be divided into two main groups [20], namely *blob tracking* and *contour tracking*, which again can be subdivided by choice of tracking algorithm into *constrained optimization* and *state estimation* categories.

This thesis focuses on the type of video tracking known as contour tracking using state estimation algorithms. Contour tracking is a form of sequential object following in video streams, where one focuses on properties of the object boundary, such as gradients and edges, instead of object interior. This differs from segmentation-based tracking approaches, known as blob tracking, which focuses properties of the object interior, such as color or texture.

Application of contour tracking using real-time sequential processing algorithms to echocardiography constitutes a novel approach, with only a handful known prior publications in existence [14], [16], [15]. All prior art is also limited to 2D echocardiography, making this thesis challenging the frontiers of the video tracking discipline within computer science.

Chapter 2

Heart Physiology

The heart [30] is a small muscular organ, whose main purpose is to pump blood through the body. It consists of four chambers, namely the *right atrium*, *right ventricle*, *left atrium* and the *left ventricle*. The atriums on each side of the heart is connected to their adjacent ventricles through valves, which opens for blood-stream in only one direction. The blood circulation tasks is divided between the two sides of the heart: Deoxygenated blood from the body is pumped through the right side of the heart, towards the lungs, and fresh oxygenated blood is pumped through the left side of the heart, towards the rest of the body.

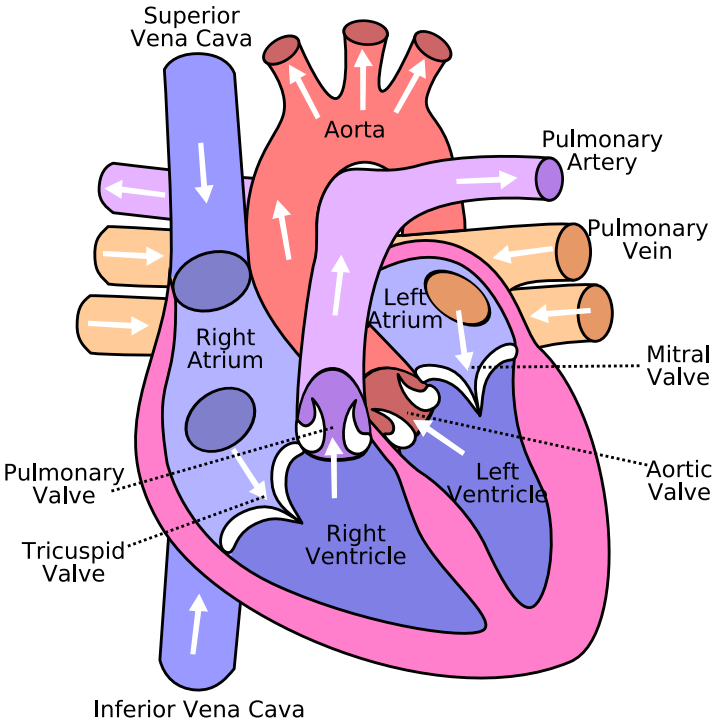


Figure 2.1: Frontal view of the heart, showing all chambers and valves. Figure courtesy of Eric Pierce.

Blood is pumped through the heart in rhythmic strokes, divided into two stages. The first

stage, *systole*, happens when the ventricles contract, leading to a decrease in chamber volume, which in turn pumps blood out of the heart. This is followed by the *diastole*, where the ventricles relax to expand, while at the same time the atriums contract, pumping new blood into the heart [26].
Two important clinical parameters, often used to diagnose the performance of the heart, are *stroke volume* and *ejection fraction*. Stroke volume is the difference between the *end diastolic volume* (EDV) and the *end systolic volume* (ESV), which is a measure of how much blood the heart pumps in each stroke. *Ejection fraction* (EF) is the ratio between the stroke volume and the end diastolic volume, measuring the percentage of the ventricle being emptied in each stroke.

$$SV = EDV - ESV$$
$$EF = \frac{SV}{EDV}$$

These measures are generally good indicators of general heart state, as cardiac diseases often leads to stiffer, less flexible hearts [29].

Chapter 3

Medical Ultrasound

Medical ultrasound, or sonography, is a diagnostic imaging modality used to visualize the interior of the body. It is based on propagating focused, high frequency sound pulses through the body, and then listening to the echo caused by internal body structures. Different organs, blood and tissue exhibit different acoustical properties, which in turn yields different echo intensities, which can be used to create images. Echo delay time can be converted into radial distance, since sound waves propagate through human tissue at approximately $1540 m/s$. The formula for calculating radial distance, $r\,[m]$, based on propagation velocity, $c\,[m/s]$, and echo delay time, $t\,[s]$, hence becomes [32]:

$$r(t) = \frac{1}{2} c\, t$$
$$r(t) = \frac{1}{2} 1540\, t$$

The division by two originates from the fact that sound waves has to travel both down into the body and be reflected back up again, before being received, leading to an effective doubling of propagation distance.

Unlike other medical imaging modalities like Magnetic Resonance Imaging (MRI) and Computed Tomography (CT), there are no radiation or strong magnetic fields involved, only sound waves. Ultrasound is therefore considered to be the safest and least harmful imaging modality used for medical diagnosis [32].

Ultrasound pulses are usually created using *phased array probes* [33], which are hand-held probes consisting of many independent piezoelectric elements arranged in an array-structure. These elements are first used to transmit pulses, by applying electrical pulses across them, which causes the crystalline structures to vibrate. Upon reception, they are also used to listen to the echoes, since acoustical vibrations induce electrical signals across the crystals.

Focused beams are created by individually delaying the pulse emitted from each element, in such a way that the sum of all pulses emitted from each element form a narrow focused beam steered in one direction. The same procedure is used upon reception to become sensitive to echoes originating from only a certain direction.

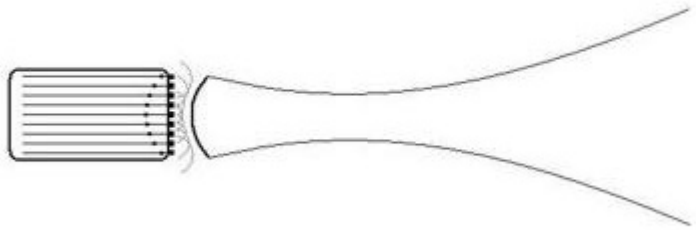


Figure 3.1: Beamforming using phased array probes in ultrasound. Pulses from the transducer elements are individually delayed to form a parabolic wavefront which contracts into a highly focused pulse at a designated depth.

3.1 Contrast Imaging

A major challenge faced by ultrasound is how acoustically homogeneous blood and tissue are. This makes it difficult to discern blood from soft tissue, like heart muscles, since the difference in backscatter intensity is quite limited. This problem is particularly challenging for patients with general poor image quality, often due to excessive body fat which also causes problems with reverberation.

Improved blood-tissue separability can be achieved by means of *contrast enhanced imaging* [27], a procedure where highly reflective microbubbles are injected into the bloodstream during ultrasound examination. These bubbles are typically filled with gas, which makes them highly compressible by ultrasound pulses. This high compressibility leads to a strong increase in backscatter intensity from the bloodstream, which in turn makes it easier to discern blood from tissue, as blood appears very bright on the images.

3.2 3D Echocardiography

Medical ultrasound can be used to image the human heart in 3D [28]. This is achieved by recording beams in fan-like structures, by steering the beams in both azimuth and elevation and stacking the scan planes upon one another, forming a volume. The resulting volume can then be visualized on a display.

The major challenge faced in 3D echocardiography is the limited propagation velocity of sound in tissue. It takes approximately $200\mu s$ for a single beam to propagate $15cm$ down through the heart and back again. This is a quite long time, since cardiac imaging requires framerates in excess of 20fps to image the heart's dynamics with sufficient temporal resolution. Together this implies that for each new frame, a maximum of 250 beams can be emitted. Volume imaging of reasonable quality requires at least a grid of 120 times 60 beams to picture the heart with sufficient detail. This number of beams is just physically impossible to achieve

without some clever thinking. Two recent innovations in 3D echocardiography helps alleviate this problem:

First, **multi-line acquisition** is used to simultaneously receive several beams per transmitted beam. This technique is based on transmitting a single, broad, beam to illuminate a designated sector within the body. Several beamformers are then used in parallel upon reception to simultaneously listen for backscatter echo in slightly different directions, effectively yielding several beams from the illuminated sector, thus achieving a significant increase in the number of beam possible per unit of time. Up to 16 parallel beamformers can be used in parallel without sacrificing too much image quality.

Secondly, **subvolume stitching** is used to record only a part of the heart, a *subvolume*, for each new frame. The remaining subvolumes are fetched from previous heartbeats, which together form a complete volume. The subvolumes are replaced in an alternating fashion, one subvolume per heartbeat. This is possible because each heartbeat looks essentially equal, so synchronized subvolumes from successive heartbeats can be combined to form a complete volume.

Together, multi-line acquisition and subvolume stitching are used to enable 3D echocardiography with both high spatial resolution and high framerate.

3.3 Scanconversion

The nature of the beamforming used in ultrasound, where focused beams are transmitted and received in directions of interest, leads to recordings originating in a spherical coordinate system, referred to as *beamspace* [34]. Recordings in this format have the following dimensions: *range* (r) for radial distance, *azimuth* (az) for rotation around the primary axis, and finally *elevation* (el) for rotation around the secondary axis.

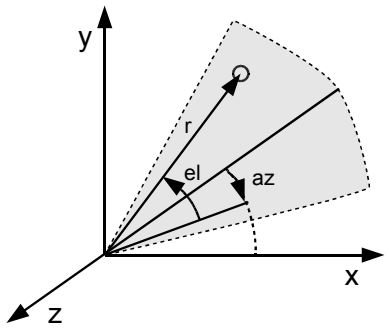


Figure 3.2: The coordinate systems used in 3D ultrasound.

Spherical coordinates can be defined in a Cartesian frame by means of the following equations:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ az &= \tan^{-1}(x/z) \\ el &= \tan^{-1}(y/\sqrt{x^2 + z^2})) \end{aligned}$$

with the image located in negative z-direction. Conversion from spherical to Cartesian coordinates, known as *scanconversion*, are required before displaying images on computers. This can be done by solving the equations above for x, y and z. This yields the following relationship:

$$\begin{aligned} x &= r\cos(el)\sin(az) \\ y &= r\sin(el) \\ z &= r\cos(el)\cos(az) \end{aligned}$$

3.4 Sample Interpolation

The acquired samples are arranged in a rectangular grid in beamspace, sampled in an equidistant fashion with constant radial-, azimuth- and elevation-distance between the samples. Samples requested from the volume are, however, rarely positioned at integral sample-coordinates, but rather on points in-between the sample grid. A weighting scheme for interpolating the intensity at the requested coordinate is therefore required.

The most commonly used technique for resolving this problem is by means of *linear interpolation* [31]. For 3D-volumes this means that the requested intensity at each coordinate is evaluated to be a weighted sum of the 8 nearest samples in the grid, based on their relative distances. The interpolated intensity at position $\{x, y, z\}$ in the discrete volume $VOL_N(...)$ is then assumed to be:

$$\begin{aligned} VOL_R(x,y,z) = {} & (1-dx)(1-dy)(1-dz)\ VOL_N(\lfloor x\rfloor\ \ \ , \lfloor y\rfloor\ \ \ , \lfloor z\rfloor\ \ \ ) \\ & + (\ \ \ dx)(1-dy)(1-dz)\ VOL_N(\lfloor x\rfloor + 1, \lfloor y\rfloor\ \ \ , \lfloor z\rfloor\ \ \ ) \\ & + (1-dx)(\ \ \ dy)(1-dz)\ VOL_N(\lfloor x\rfloor\ \ \ , \lfloor y\rfloor + 1, \lfloor z\rfloor\ \ \ ) \\ & + (\ \ \ dx)(\ \ \ dy)(1-dz)\ VOL_N(\lfloor x\rfloor + 1, \lfloor y\rfloor + 1, \lfloor z\rfloor\ \ \ ) \\ & + (1-dx)(1-dy)(\ \ \ dz)\ VOL_N(\lfloor x\rfloor\ \ \ , \lfloor y\rfloor\ \ \ , \lfloor z\rfloor + 1) \\ & + (\ \ \ dx)(1-dy)(\ \ \ dz)\ VOL_N(\lfloor x\rfloor + 1, \lfloor y\rfloor\ \ \ , \lfloor z\rfloor + 1) \\ & + (1-dx)(\ \ \ dy)(\ \ \ dz)\ VOL_N(\lfloor x\rfloor\ \ \ , \lfloor y\rfloor + 1, \lfloor z\rfloor + 1) \\ & + (\ \ \ dx)(\ \ \ dy)(\ \ \ dz)\ VOL_N(\lfloor x\rfloor + 1, \lfloor y\rfloor + 1, \lfloor z\rfloor + 1) \end{aligned}$$

where the weights are defined to be

$$\{dx, dy, dz\} = \{x - \lfloor x\rfloor,\ y - \lfloor y\rfloor,\ z - \lfloor z\rfloor\} \in [0, 1)$$

It can easily proven that the weight-coefficients always sum up to one, and $VOL_R(x, y, z) = VOL_N(x, y, z)$ when $\{x, y, z\}$ are integers.

Chapter 4

State Space Modeling

A state space model is a mathematical model of a process, where the process' *state* ($\mathbf{x}$) is represented by a numerical vector. State-space models usually consists of two separate models: a kinematic *process model*, which describes how the state propagates in time based on external influences, such as input and noise; and a *measurement model*, which describe how measurements ($\mathbf{z}$) are taken from the process, typically simulating noisy and/or inaccurate measurements.

4.1 General State Space Model

The most general form of state-space models is the nonlinear model. This model can be described using two functions, f and h:

$$\mathbf{x}_{k+1} = f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k)$$
$$\mathbf{z}_k = h(\mathbf{x}_k, \mathbf{v}_k)$$

which govern state propagation and measurements, respectively. $\mathbf{u}$ is process input, and $\mathbf{w}$ and $\mathbf{v}$ are stochastic state and measurement noise vectors, respectively while k is the discrete time.

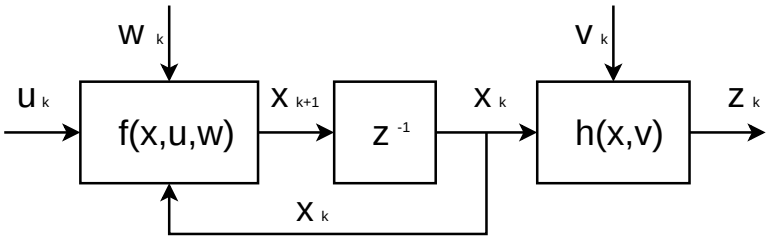


Figure 4.1: *A general state-space model. z^{-1} is the unit delay function known from the Z-transform in digital signal processing.*

State-space models are remarkably usable for modeling almost all sorts of processes. f and h are usually based upon a set of discretized differential equations, governing the dynamics of and observations from the process.

4.2 Linear State Space Model

A linear state-space model is a model where the functions f and h are linear in both state and input. The functions can then be expressed by using the matrices $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{H}$, reducing state propagation calculations to linear algebra. Overall this results in the following linear state-space model:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \end{aligned}$$

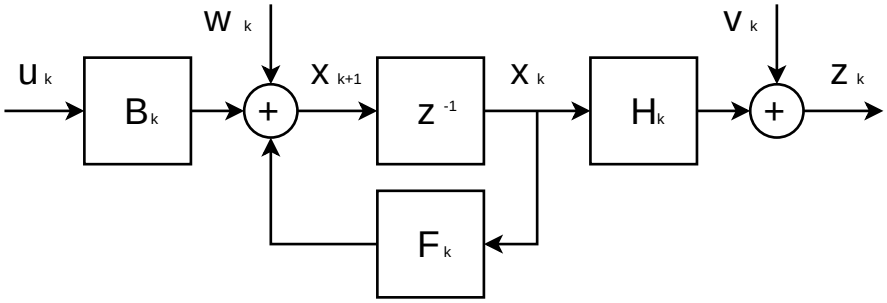


Figure 4.2: *A linear state-space model*

This linear model is easier both to calculate and analyze, enabling modelers to investigate properties such as controllability, observability and frequency response [10]. Linear state models are either based on inherently linear processes, or simply a linearized version of a nonlinear process by means of a first order Taylor approximation.

Chapter 5

Sequential State Estimation

State estimation concerns the problem of estimating the *probability density function* (pdf) for the state of a process which is not directly observable. This typically involves both predicting the next state based on the current, and updating/correcting this prediction based on noisy measurements taken from the process.

5.1 Recursive Bayesian Estimation

The most general form for state estimation is known as recursive Bayesian estimation [2]. This is the optimal way of predicting a state pdf for any process, given a system and a measurement model. In this section we will discuss this estimator, which recursively calculates a new estimate for each time-step, based on the estimate for the previous timestep and new measurements.

Recursive Bayesian estimation works by simulating the process, while at the same time adjusting it to account for new measurements $\mathbf{z}$, taken from the real process. The calculations are performed recursively in a two step procedure. First, the next state is predicted, by extrapolating the current state onto next time step using state propagation belief $p(\mathbf{x}_k|\mathbf{x}_{k-1})$ obtained from function f. Secondly, this prediction is corrected using measurement likelihood $p(\mathbf{z}_k|\mathbf{x}_k)$ obtained from function h, taking new measurements into account.

The Chapman-Kolmogorov equation is first used to calculate a prior pdf for state $\mathbf{x}_k$, based on measurements up to time $k-1$:

$$p(\mathbf{x}_k|\mathbf{z}_{k-1}) = \int p(\mathbf{x}_k|\mathbf{x}_{k-1})p(\mathbf{x}_{k-1}|\mathbf{z}_{k-1})\, d\mathbf{x}_{k-1}$$

Bayes rule is then used to calculate the updated pdf for state $\mathbf{x}_k$, after taking measurements up to time k into account:

$$p(\mathbf{x}_k|z_k) = \frac{p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{k-1})}{\int p(\mathbf{z}_k|\mathbf{x}_k)p(\mathbf{x}_k|\mathbf{z}_{k-1})\, d\mathbf{x}_k}$$

Unfortunately, this method does not scale very well in practice, mainly due to the large state space for multidimensional state vectors. Calculating the prior probability of each point in

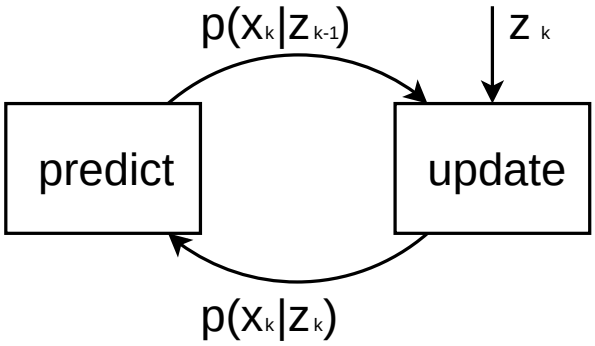


Figure 5.1: *Recursive Bayesian estimator loop*

this state space involves a multidimensional integral, which quickly becomes intractable as the state space grows. Computers are also limited to calculation of the pdf in discrete point in state space, requiring a discretization of the state space. This technique is therefore mainly considered as a theoretic foundation for state estimation in general. Bayesian estimation by means of computers is only possible if either the state space can be discretized efficiently, or if certain limitations apply for the model.

5.2 Kalman Filter

The problem of state estimation can be made tractable if we put certain constrains on the system models, by requiring both f and h to be linear functions, and the noise terms $\mathbf{w}$ and $\mathbf{v}$ to be uncorrelated, Gaussian and white with zero mean. Put in mathematical notation, we then have the following constraints:

$$\begin{aligned} f(\mathbf{x}_k, \mathbf{u}_k, \mathbf{w}_k) &= \mathbf{F}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \\ h(\mathbf{x}_k, \mathbf{v}_k) &= \mathbf{H}\mathbf{x}_k + \mathbf{v}_k \\ \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{Q}_k) \qquad & \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{R}_k) \\ E(\mathbf{w}_i\mathbf{w}_j^T) = \mathbf{Q}_i\delta_{i-j} \qquad & E(\mathbf{v}_i\mathbf{v}_j^T) = \mathbf{R}_i\delta_{i-j} \\ E(\mathbf{w}_k\mathbf{v}_k^T) &= \mathbf{0} \end{aligned}$$

where $\mathbf{Q}$ and $\mathbf{R}$ are covariance matrices, describing the second-order properties of the state- and measurement noise. The constraints described above reduce the state model to:

$$\begin{aligned} \mathbf{x}_{k+1} &= \mathbf{F}_k\mathbf{x}_k + \mathbf{B}_k\mathbf{u}_k + \mathbf{w}_k \\ \mathbf{z}_k &= \mathbf{H}_k\mathbf{x}_k + \mathbf{v}_k \end{aligned}$$

where $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{H}$ are matrices, possible time dependent.

As the model is linear and input is Gaussian, we know that the state and output will also be Gaussian [25]. The state and output pdf will therefore always be normally distributed, where mean and covariance are sufficient statistics. This implies that it is not necessary to calculate a full state pdf any more, a mean vector $\hat{\mathbf{x}}$ and covariance matrix $\hat{\mathbf{P}}$ for the state will suffice.

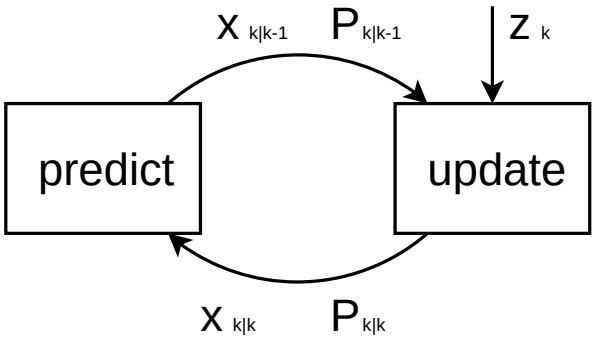


Figure 5.2: *Kalman filter loop*

The recursive Bayesian estimation problem is then reduced to the *Kalman filter* [12], where f and h are replaced by the matrices $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{H}$. The Kalman filter is, just as the Bayesian estimator, decomposed into two steps: predict and update. The actual calculations required are:

Predict next state, before measurements are taken:

$$\begin{aligned}\bar{\mathbf{x}}_{k|k-1} &= \mathbf{F}_k\hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k\mathbf{u}_k \\ \bar{\mathbf{P}}_{k|k-1} &= \mathbf{F}_k\hat{\mathbf{P}}_{k-1|k-1}\mathbf{F}_k^T + \mathbf{Q}_k\end{aligned}$$

Update state, after measurements are taken:

$$\begin{aligned}\mathbf{K}_k &= \bar{\mathbf{P}}_{k|k-1}\mathbf{H}_k^T(\mathbf{H}_k\bar{\mathbf{P}}_{k|k-1}\mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \hat{\mathbf{x}}_{k|k} &= \bar{\mathbf{x}}_{k|k-1} + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}_k\bar{\mathbf{x}}_{k|k-1}) \\ \hat{\mathbf{P}}_{k|k} &= (\mathbf{I} - \mathbf{K}_k\mathbf{H}_k)\bar{\mathbf{P}}_{k|k-1}\end{aligned}$$

where $\mathbf{K}$ is the Kalman gain matrix, used in the update observer, and $\hat{\mathbf{P}}$ is the covariance matrix for the state estimate, containing information about the accuracy of the estimate. More details and background for this filter can be found in [9]. The Kalman filter is quite easy to calculate, due to the fact that it is mostly linear, except for a matrix inversion. It can also be proved that the Kalman filter is an optimal estimator of process state in a *minimum mean square error* (MMSE) sense [9], meaning it minimizes the expected quadratic error.

5.3 Extended Kalman Filter

Most processes in real life are unfortunately not linear, and therefore needs to be linearized before they can be estimated by means of a Kalman filter. The extended Kalman filter (EKF) [12] solves this problem by calculating the Jacobian[1] of f and h around the estimated state, which in turn yields a trajectory of the model function centered on this state.

[1]The Jacobian is the matrix of all partial derivatives of a vector with regards to the state vector.

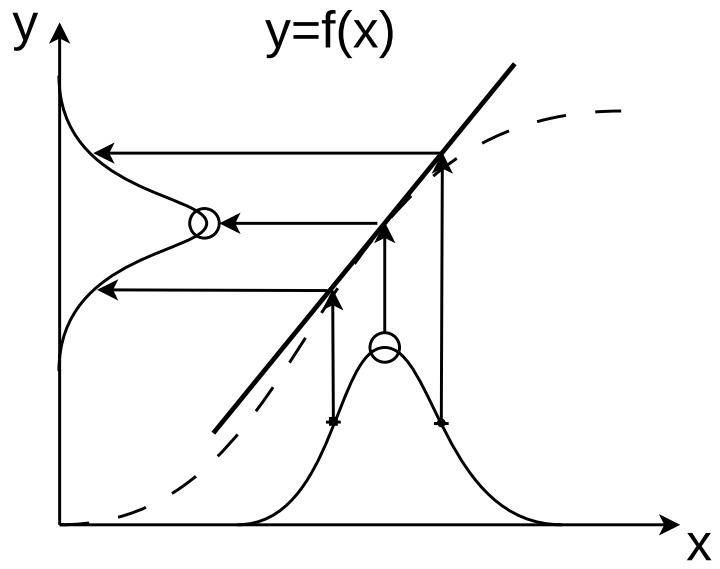


Figure 5.3: *Illustration of how the Extended Kalman filter linearizes a nonlinear function around the mean of a Gaussian distribution, and thereafter propagates the mean and covariance through this linearized model*

$$\mathbf{F}_k = \mathbf{J}_{\mathbf{x}} f(\mathbf{x}, \mathbf{u}, \mathbf{w}) = \left. \frac{\partial f(\mathbf{x}, \mathbf{u}, \mathbf{w})}{\partial \mathbf{x}} \right|_{\hat{\mathbf{x}}_{k|k}, \mathbf{u}_k, 0}$$
$$\mathbf{H}_k = \mathbf{J}_{\mathbf{x}} h(\mathbf{x}, \mathbf{v}) = \left. \frac{\partial h(\mathbf{x}, \mathbf{v})}{\partial \mathbf{x}} \right|_{\bar{\mathbf{x}}_{k|k-1}, 0}$$

The extended Kalman filter works almost like a regular Kalman filter, except for $\mathbf{F}$ and $\mathbf{H}$, which vary in time based on the estimated state $\hat{\mathbf{x}}$. The actual calculations required are:

Predict next state, before measurements are taken:

$$\bar{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k, 0)$$
$$\bar{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{P}}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

Update state, after measurements are taken:

$$\mathbf{K}_k = \bar{\mathbf{P}}_{k|k-1} \mathbf{H}_k^T (\mathbf{H}_k \bar{\mathbf{P}}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k)^{-1}$$
$$\hat{\mathbf{x}}_{k|k} = \bar{\mathbf{x}}_{k|k-1} + \mathbf{K}_k (\mathbf{z}_k - h(\bar{\mathbf{x}}_{k|k-1}, 0))$$
$$\hat{\mathbf{P}}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \bar{\mathbf{P}}_{k|k-1}$$

where $\mathbf{K}$ is the Kalman gain matrix, used in the update observer, and $\mathbf{P}$ is the covariance matrix for the state estimate, containing information about the accuracy of the estimate.

5.4 Information Filter

There exists an alternative formulation of the discrete Kalman filter, namely the *information filter*. This form, which is algebraically equal to the standard form, is often advantageous in situations where the number of measurements exceed the state dimension, since it avoids the problem of inverting matrices of the size of the measurement covariance. The information filter also enables efficient processing when many independent measurements are taken at each timestep [9].

Just like the standard Kalman filter, this variant starts by predicting the state at the next timestep. The equations are equal to those above:

$$\bar{\mathbf{x}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{x}}_{k-1|k-1} + \mathbf{B}_k \mathbf{u}_k$$
$$\bar{\mathbf{P}}_{k|k-1} = \mathbf{F}_k \hat{\mathbf{P}}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k$$

The measurement update equations, however, are quite different. They are expressed on a *summing form*, where prior knowledge from the prediction is added to the knowledge from the measurements:

$$\hat{\mathbf{P}}_{k|k}^{-1} = \bar{\mathbf{P}}_{k|k-1}^{-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k$$
$$\hat{\mathbf{P}}_{k|k}^{-1} \hat{\mathbf{x}}_{k|k} = \bar{\mathbf{P}}_{k|k-1}^{-1} \bar{\mathbf{x}}_{k|k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k$$

Inversion of the resulting sums are of course required to acquire the updated state estimate and covariance:

$$\hat{\mathbf{P}}_{k|k} = (\bar{\mathbf{P}}_{k|k-1}^{-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{H}_k)^{-1}$$
$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{P}}_{k|k} (\bar{\mathbf{P}}_{k|k-1}^{-1} \bar{\mathbf{x}}_{k|k-1} + \mathbf{H}_k^T \mathbf{R}_k^{-1} \mathbf{z}_k)$$

and by expanding the last equation above and comparing it to the original Kalman equations we discover and alternative form of the Kalman gain, namely;

$$\mathbf{K}_k = \hat{\mathbf{P}}_{k|k} \mathbf{H}_k^T \mathbf{R}_k^{-1}$$

This form collects everything related to the measurements into $\mathbf{H}^T \mathbf{R}^{-1} \mathbf{H}$ and $\mathbf{H}^T \mathbf{R}^{-1} \mathbf{z}$, which are known as the *information matrix*, and *information vector*, respectively.

5.4.1 Multiple Independent Measurements

One often encounters situations where several independent measurements are taken at each time step. The information filter formulation of the Kalman filter can often be computationally more efficient in such situations, as will be shown below.

The calculation of information-matrices and -vectors for independent measurements yield equations involving inversion of the diagonal measurement covariance matrix. The structure

of the equations can then be exploited to convert it into efficient summations:

$$\begin{aligned}
\mathbf{H}^T\mathbf{R}^{-1}\mathbf{z} &= \begin{bmatrix} | & | & | \\ \mathbf{h}_1 & \cdots & \mathbf{h}_N \\ | & | & | \end{bmatrix} \begin{bmatrix} r_1^{-1} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & r_N^{-1} \end{bmatrix} \begin{bmatrix} z_1 \\ . \\ z_N \end{bmatrix} \\
&= \sum_i \mathbf{h}_i r_i^{-1} z_i \\
\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} &= \begin{bmatrix} | & | & | \\ \mathbf{h}_1 & \cdots & \mathbf{h}_N \\ | & | & | \end{bmatrix} \begin{bmatrix} r_1^{-1} & 0 & 0 \\ 0 & \cdots & 0 \\ 0 & 0 & r_N^{-1} \end{bmatrix} \begin{bmatrix} - & \mathbf{h}_1^T & - \\ - & \cdots & - \\ - & \mathbf{h}_N^T & - \end{bmatrix} \\
&= \sum_i \mathbf{h}_i r_i^{-1} \mathbf{h}_i^T
\end{aligned}$$

This formulation replaces the expensive measurement covariance matrix inversion with simple scalar divisions. It also involves fewer multiplications. The principal advantage of this formulation, is that it makes it conceptually simple to fuse information from multiple independent measurements together, simply be summing information-matrices and -vectors together.

Part II

Methods

Chapter 6

Contour Tracking Framework

This thesis does, as pointed out in the introduction, follow a contour tracking using state estimation algorithms approach to video tracking. Contour tracking is a form of sequential object following in video streams, where one focuses on properties of the object boundary, such as gradients and edges, instead of object interior. This differs from segmentation-based tracking approaches, known as blob tracking, which focuses properties of the object interior, such as color or texture.

6.1 Contour Versus Blob Tracking

The principal advantage of contour tracking compared to segmentation-based tracking is reduced computational complexity, because only the object boundary needs to be examined, and not the entire interior. This does, however, require objects with clearly present edges, that can be used to detect the presence of an object. Together, this results in what is primarily an edge-detection problem, instead of a segmentation problem.

Edges are typically present at object boundaries in images, which make them detectable. Contour tracking utilizes edge-detection algorithms to search for edges present in proximity to the contour. This is often done by searching for edges in the normal direction of a predicted contour at regularly spaced intervals. The resulting distance between contour and edge is referred to as *normal displacement*, and can be used to correct the prediction.

6.2 State Estimation Approach

This thesis follows a *sequential state estimation* approach to contour-tracking. This differs from the more common *constrained optimization* approach where the tracking problem is modeled as a cost minimization problem, and iteratively refined using gradient descent algorithms. This choice is primarily motivated by the goal of real-time running time, since optimization approaches are reported to be requiring hundreds of iterations to converge [1], leading to running time in excess of several minutes for accurate tracking throughout the heart cycle [35]. A such comparison is, however, not entirely fair, since iterative refinement algorithms are believed to yield more accurate tracking compared to their estimation counterparts.

The state estimation approach is centered around a *state vector*, which contains the contour state parameters. This vector can either be a concatenation of all contour vertex coordinates, as is done in [19] and [22], or it can be input to a deformation model that relates state vector to contour appearance. The latter approach, which is often referred to as *subspace modeling* of *deformable contours*, is pursued here, since it enables contour representation using far fewer parameters compared to the concatenated vertex approach. This is especially critical for tracking in 3D, since contour surface models typically require several hundred vertices or more for accurate surface representation, leading to problems with an excessive number of freedoms. Usage of deformation models also opens up for incorporation of prior knowledge by constraining the deformation model to shapes similar to the objects being tracked, as well as having intuitive and meaningful tracking parameters, such as translation, rotation and scaling.

Contour tracking can then be modeled as a *sequential state estimation* problem with prediction, measurement and update steps, where the goal is to estimate the state of the deformation model. Prediction is performed using a kinematic model that predicts contour appearance in next frame based on current estimates. Measurements can similarly be based on normal displacement measurements and a model that relates edge measurement to changes in contour state. This modeling framework then enables usage of algorithms from the field of statistical state estimation, such as the Kalman filter and particle filters.

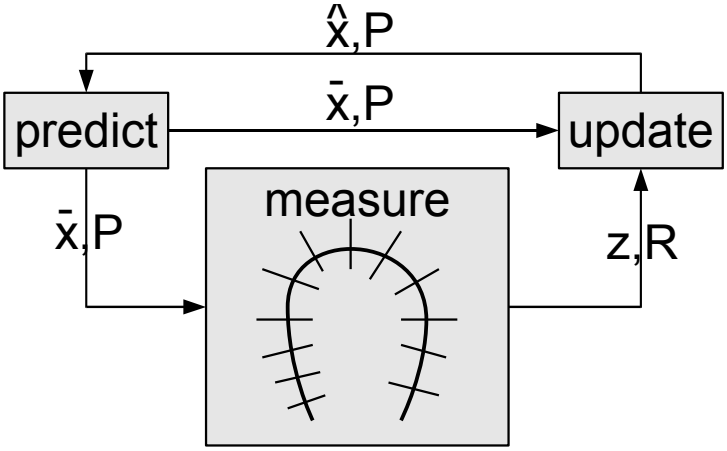


Figure 6.1: Sequential contour tracking dataflow.

Altogether, this results in a tracking framework that follows the following program flow:

1. Predict state in next frame, using a kinematic model.
2. Use a deformation model to create a predicted contour based on an initial contour and the predicted state.
3. Detect edges along contour normals on the predicted contour.
4. Update the state prediction using the edge measurements.

This state estimation approach was first presented by Blake et al. in [6], [8] and [7] for B-spline models deformed by linear transforms. Later, the framework was used for real-time

left ventricular tracking in long-axis 2D-echocardiography by Jacob et al. in [14], [16] and [15], which all used a B-spline representation, deformed by linear transforms, using a trained *principal component analysis* (PCA) deformation model. All these papers also discussed the possibility of extending the framework to 3D-echocardiography, which is what has been done in this thesis.

6.3 Limitations

Kalman based tracking is inherently limited to estimating unimodal distribution. It is local, and will therefore lock on to the best and nearest object with a reasonable fit. A Kalman based framework is therefore not suitable for tracking in a cluttered environment.

Robust tracking can nevertheless be accomplished by extending the framework with multiple-hypothesis tracking, using sequential Monte Carlo methods [2]. This opens up the possibility of both tracking in cluttered environments, as well as tracking of multiple simultaneous objects. This thesis will, however, not pursuit this strategy, as it will be in focus for future research within this field.

6.4 Relationship to Sampling-based Contour Tracking

The *condensation algorithm* presented by Isard and Blake in [13] uses a statistical sampling-based scheme, known as Sampling Importance Resampling (SIR), to track deformable contours in video streams. This algorithm uses edge-measurements merely to evaluate contour likelihood, and does not use them to correct the predictions, as is done in Kalman-based tracking. This leads to tracking that is undoubtedly more robust in the presence of clutter. But it is, however, much more computationally demanding due to the requirement for often several hundred parallel particles in order to get good tracking results [13], making it unsuitable for real-time implementations. It is also much easier to implement, primarily because the measurements are solely used to evaluate contour likelihood.

6.5 Framework Decomposition

The proposed tracking approach results in a very modular framework, with modules responsible for clearly separated tasks within the framework. These modules have clearly defined interfaces, independent of internal implementation, which makes it easy to compare different models and algorithms without altering the overall framework.

The framework can be divided into the 5 different modules. A list over the modules, with examples of model and/or algorithmic content suitable for implementation might then include:

1. **Tracking algorithm**
   (a) Extended Kalman Filter *[implemented & used]*
   (b) Iterated Kalman filter [3]
   (c) *Condensation* algorithm (sampling importance resampling particle filter)
   (d) Particle filter with Kalman-based proposal function

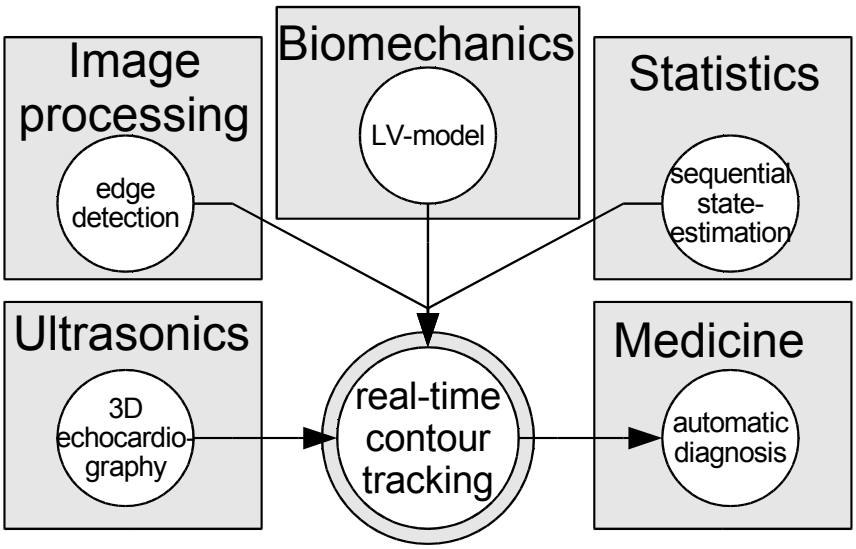


Figure 6.2: Modular overview over the tracking framework

2. **Contour model**
   (a) Polygon *[implemented]*
   (b) Spline *[implemented]*
   (c) Parametric surface
   (d) Set of points, with associated normal vectors *[implemented & used]*
   (e) Other contour representations
3. **Deformation model**
   (a) Rigid deformations *[implemented]*
   (b) Affine deformations *[implemented]*
   (c) PCA-based deformation models
   (d) General, nonlinear deformations *[implemented & used]*
4. **Edge model**
   (a) Gradient based models
   (b) Intensity "step" model *[implemented & used]*
   (c) Intensity "peak" model *[implemented]*
   (d) Other edge models
5. **Input data**
   (a) 2D video (television, computer camera, etc.) *[implemented]*
   (b) 2,5D video - depth maps (stereo vision)

(c) 3D video (3D MRI and Echocardiography) *[implemented & used]*

The models and algorithms in the list above are flagged to indicate wether they have acutally been implemented, and if they are also used conjunction to this thesis.

6.6 Notable Special Cases

The framework presented here is basically a generalization of the *Kalman-based* and *Condensation* contour tracking algorithms presented by Blake and Isard in [7] and [13], with their algorithms listed below as special cases of the framework:

Blake's Kalman-filter based contour tracking: Kalman-filter based tracking of spline-contours deformed by linear deformations using a gradient-based edge detector.

Condensation algorithm: Sampling importance resampling (SIR) particle filter tracking of spline-contours deformed by linear deformations using a gradient-based edge detector.

Proposed LV-tracking framework: Extended Kalman-filter based tracking of parametric surfaces deformed by nonlinear deformations using a "step" based edge model.

6.7 Outline for the Following Chapters

The remaining chapters in the *methods*-part of this thesis will present the inner workings of all aspects of the contour tracking framework. To summarize, the following chapters will discuss:

Measurement Models: Presents how contour and deformation models relate to the tracking algorithm.

Left Ventricle Model: Presents the contour and deformation model used.

Kinematic Model: Presents the kinematic aspect of the tracking algorithm.

Kalman Filter Equations for Tracking: Presents details on the extended Kalman Filter (EKF) tracking algorithm.

Edge Detection: Presents edge detection algorithms

In total, these chapters covers the content of every module within the tracking framework except *input data*, which is considered outside the scope of this thesis, and partly covered by the *scanconversion* section in the introduction.

Chapter 7

Measurement Models

This chapter presents one of the main contributions to this thesis, namely the support for contours with general, nonlinear modes of deformation. This separates the thesis from existing *shape space* literature from Blake and Jacob [6],[8],[7],[14], [16] and [15], which were all limited to contours deformed by linear deformations, such as affine, or trained PCA-models. The choice of *normal displacement* measurements leads to results that are dimensionality independent, thus functioning just as well for tracking in 3D datasets as in 2D datasets.

Sampling-based approaches to contour tracking, such as methods based on the *condensation algorithm* [13], have previously used nonlinear deformation models, but these were all inherently simpler, due to the usage of measurements solely for evaluating contour likelihood. No known attempt has been made to improve the predicted contour deformations based on measurements for nonlinear deformation models.

Another contribution is the establishment of a clear connection to state estimation theory, by basing all edge measurements on a mathematical measurement model based on the deformation model. Previous art have been a bit vague on this point [7], by often focusing more on the resulting algorithms than their relationship to stochastic modeling and estimation. Important aspects, such as linearization of nonlinear deformation models and proper handling of measurement uncertainty can therefore be handled in a consistent and intuitive way, based on established practice from estimation theory. Altogether, this leads to an elegant yet powerful framework, without any ad-hoc solutions.

To summarize, the main contributions of this chapter are:

1. Generalization of Blake's framework to nonlinear deformation models
2. Extension of Blake's framework to tracking in 3D
3. Establishment of a clear connection to estimation theory

7.1 Contour Models

Contour models are used to represent and parameterize contours, and thus act as templates for the objects being tracked. The framework supports many forms of contours, such as polygons,

quadratic and cubic B-splines, implicit curves and parametric surfaces among others. The only requirement is that it must be possible to evaluate the position and normal vector of regularly spaced points on the contour.

7.1.1 Point Sets

The simplest and most general case of contour modeling is by means of point sets. The contour then simply consists of a set of vertices with associated normal vectors. These points will typically originate from an underlying geometric model.

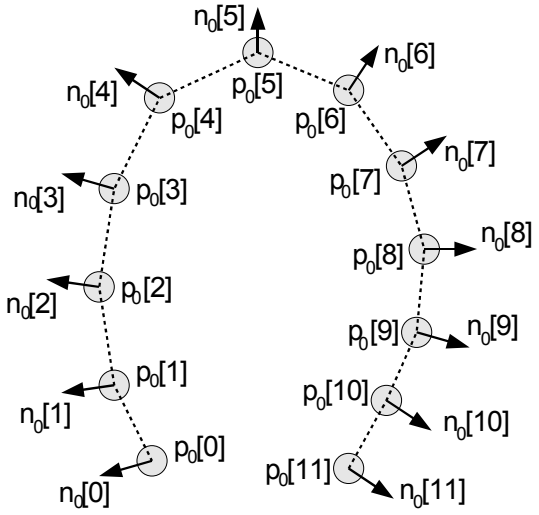


Figure 7.1: Example initial contour, consisting of a set of vertices with associated normals.

7.1.2 B-splines

Another approach, especially suitable for contour modeling in 2D, is B-splines. The contour are then parameterized by a set of control points. This results in smooth contours that can be modified intuitively by adjusting the control points. Normal vectors can easily be be derived automatically. Prior work by Blake and Jacob have followed this approach.

7.2 Deformation Models

The goal of using deformation models is to model the freedoms of deformation that objects being tracked has. This effectively both regularizes the allowable contours and reduces tracking dimensionality. The degrees of freedom typically involve global deformations, such as translation, rotation and scaling, but also local deformations for controlling changes of appearance. Altogether, this makes the deformation model one of the most important part of the tracking framework, as it both models and restricts the allowable deformations for objects.

Unlike previous work, this framework places very few restrictions on the allowable deformation models. Virtually "any" deformation model can be used, as long as it can be expressed mathematically. One must, however, be able to derive the partial derivatives of the position

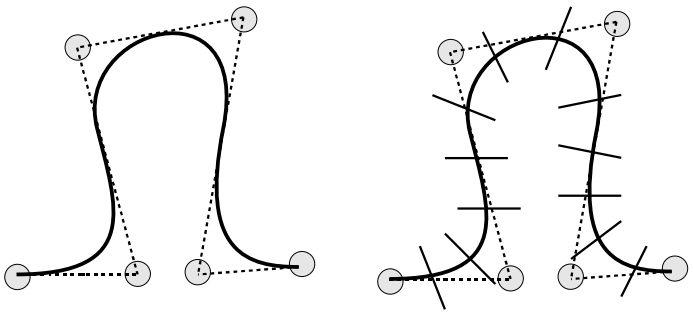


Figure 7.2: Example B-spline contour, with and without normals. The grey circles are the control points used to alter the contour' appearance.

as a function of the deformation parameters. The general deformation model form chosen is that of a function $\mathcal{D}(...)$ that transforms initial contour coordinates into deformed coordinates using a state vector as parameter:

$$\mathbf{p} = \mathcal{D}(\mathbf{p}_0, \mathbf{x})$$

This differs from the linear *shape space* deformation models used by Blake and Jacob [6],[8],[7],[14], [16] and [15], where all deformations had to be on the following form:

$$\mathbf{p} = \mathbf{p}_0 + \mathbf{W}\mathbf{x}$$

where $\mathbf{W}$ is a deformation matrix which restricts the allowable deformation models to transforms that are linear in the state vector, such as affine and PCA-models.

7.2.1 Normal Deformations

Whereas propagation of vertices through the deformation model is trivial, deformation of the associated normal vectors through the deformation model is a bit trickier, since direct propagation of normal vectors will not yield correct results. A more thorough approach is therefore needed.

A review of existing literature [5] suggests that a linearization of the deformation model is also needed here, just as for the measurement vectors. The main difference is that whereas processing of measurement vectors require a shape-space Jacobian, normal deformation require a spatial Jacobian, consisting of all partial derivatives of the deformation model with regards to the input vertex dimensions.

The spatial Jacobian matrix for the deformation model is therefore defined to be:

$$\mathbf{J}_{\mathbf{p}}\mathcal{D}(\mathbf{p}_0, \mathbf{x}) = \frac{\partial \mathcal{D}(\mathbf{p}_0, \mathbf{x})}{\partial \mathbf{p}} = \begin{bmatrix} \frac{\partial \mathcal{D}_x}{\partial x} & \frac{\partial \mathcal{D}_x}{\partial y} & \frac{\partial \mathcal{D}_x}{\partial z} \\ \frac{\partial \mathcal{D}_y}{\partial x} & \frac{\partial \mathcal{D}_y}{\partial y} & \frac{\partial \mathcal{D}_y}{\partial z} \\ \frac{\partial \mathcal{D}_z}{\partial x} & \frac{\partial \mathcal{D}_z}{\partial y} & \frac{\partial \mathcal{D}_z}{\partial z} \end{bmatrix}_{\mathbf{p}=\mathbf{p}_0}$$

Correct deformation of normal vectors is accomplished by multiplying the transpose of the inverse spatial Jacobian with the initial normal vector, followed by a renormalization of the resulting vector:

$$\mathbf{n} = normalize\left\{(\mathbf{J_p}\mathcal{D}(\mathbf{p}_0, \mathbf{x}))^{-T}\mathbf{n}_0\right\}$$

Derivation of this formula, as well as proof of correctness can be found in Barr [5].

7.3 Edge Measurements

Edge measurements are used in the measurement model for contour tracking. The distance between predicted contour and measured edge, called *normal-displacement* forms the measurements, which are defined in the measurement model.

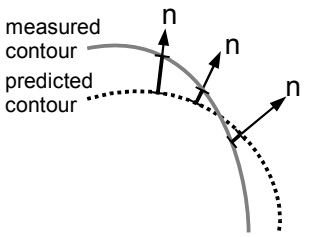


Figure 7.3: Illustration of normal displacement measurements

The actual measurements are based on the normal-displacement of contour points:

$$\begin{aligned} z_i &\equiv \mathbf{n}^T\mathbf{p}_{obs} \\ &= \mathbf{n}_i^T\mathcal{D}(\mathbf{p}_i, \mathbf{x}) + v_i \end{aligned}$$

where $\mathbf{n}$ is a unit normal vector ($|\mathbf{n}| \equiv 1$) for the contour point, and the measurement noise $v_i \sim N(0, R)$ is the measured normal displacement. The normal displacement (innovation) for each measurement then becomes:

$$v_i = \mathbf{n}^T(\mathbf{p}_{obs} - \mathcal{D}(\mathbf{p}_i, \mathbf{x}))$$

Measurement noise R is typically assumed to be constant for all edges, but can also be dependent on edge-strength or other measure of uncertainty.

The inner-product form of the normal displacement equation has the nice property of being dimensionally invariant. Contour tracking can therefore naturally be extended to 3D by simply extending the point and normal vectors with a third dimension. Normal displacement measurements in 2D will typically look like:

$$v_i = \begin{bmatrix}\mathbf{n}_x & \mathbf{n}_y\end{bmatrix}\left(\begin{bmatrix}\mathbf{p}_{obs,x} \\ \mathbf{p}_{obs,y}\end{bmatrix} - \begin{bmatrix}\mathcal{D}(\mathbf{p}_i, \mathbf{x})_x \\ \mathcal{D}(\mathbf{p}_i, \mathbf{x})_y\end{bmatrix}\right)$$

while in 3D they become:

$$v_i = \begin{bmatrix}\mathbf{n}_x & \mathbf{n}_y & \mathbf{n}_z\end{bmatrix}\left(\begin{bmatrix}\mathbf{p}_{obs,x} \\ \mathbf{p}_{obs,y} \\ \mathbf{p}_{obs,z}\end{bmatrix} - \begin{bmatrix}\mathcal{D}(\mathbf{p}_i, \mathbf{x})_x \\ \mathcal{D}(\mathbf{p}_i, \mathbf{x})_y \\ \mathcal{D}(\mathbf{p}_i, \mathbf{x})_z\end{bmatrix}\right)$$

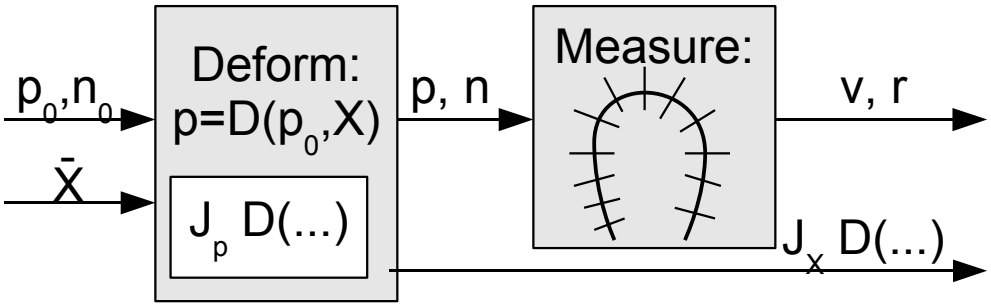


Figure 7.4: Detailed overview over the contour deformation and edge measurement process. The figure shows how the initial contour ($\mathbf{p}_0$, $\mathbf{n}_0$) is first deformed using a predicted state ($\bar{\mathbf{x}}$), yielding a deformed contour ($\mathbf{p}$, $\mathbf{n}$) and a state-space Jacobian ($\mathbf{J}_\mathbf{x}\ \mathcal{D}(\mathbf{p}_0, \mathbf{x})$). Edges are then measured relative to the predicted contour, resulting in normal displacements (v) with associated measurement error variances (r).

7.4 Measurement Linearization

Measurement vectors relate changes contour in position through normal displacement measurements to changes in deformation state of the contour. These vectors do therefore play an essential role in contour tracking. Inference of measurement matrices for linear deformations, as done in previous work [7], was trivial, since all deformation models then could be expressed in linear algebra, using matrices.

Measurement vectors for nonlinear deformations can unfortunately not be inferred directly from the models. Instead, the models have to be linearized around the predicted deformation state, using an *extended Kalman filter* approach, previously known from state-space models with nonlinear measurements in estimation theory. The linearized measurement models can then be expressed as a vector, and used in the same way in estimation algorithms as measurement vectors from inherently linear models.

Altogether, this results in a measurement vector $\mathbf{h}$ that is based on the state-space Jacobian of the measurement model, meaning all the partial derivatives of contour position, with regard to all state dimensions, evaluated at the predicted state.

The state-space Jacbian of the measurement model is defined to be the matrix consisting of the following partial derivatives:

$$\mathbf{J}_\mathbf{x}\mathcal{D}(\mathbf{p}_0, \mathbf{x}) = \frac{\partial \mathcal{D}(\mathbf{p}_0, \mathbf{x})}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial \mathcal{D}_x}{\partial x_1} & \frac{\partial \mathcal{D}_x}{\partial x_2} & \cdots & \frac{\partial \mathcal{D}_x}{\partial x_N} \\ \frac{\partial \mathcal{D}_y}{\partial x_1} & \frac{\partial \mathcal{D}_y}{\partial x_2} & \cdots & \frac{\partial \mathcal{D}_y}{\partial x_N} \\ \frac{\partial \mathcal{D}_z}{\partial x_1} & \frac{\partial \mathcal{D}_z}{\partial x_2} & \cdots & \frac{\partial \mathcal{D}_z}{\partial x_N} \end{bmatrix}$$

The linearized measurement vector thus becomes the normal displacement projection of this Jacobian matrix:

$$\mathbf{h}^T \equiv \mathbf{n}^T \mathbf{J}_\mathbf{x}\mathcal{D}(\mathbf{p}_0, \mathbf{x})$$

7.5 Measurement Processing

The assumption of independent measurements allows measurements to be summed together efficiently in *information space* using the information filter formulation for measurement processing. Independent measurements leads to a diagonal measurement covariance matrix, which in turn makes the resulting measurements a linear sum of simple vector products. No matrix products or matrix inversions are therefore required for processing.

Further simplifications to the processing is accomplished by summing the measurement innovations, being the normal-displacements, instead of the actual edge-measurement coordinates. This alleviates the Kalman filter from subtracting the predicted measurement value from the measurements in the measurement update step.

The combination of independent measurements, and innovation-processing leads to measurement processing on the following form:

$$\mathbf{H}^T\mathbf{R}^{-1}\mathbf{v} = \sum \mathbf{h}_i r_i^{-1} v_i$$
$$\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H} = \sum \mathbf{h}_i r_i^{-1} \mathbf{h}_i^T$$

which is the same form as Blake & Isard used in "Active Contours" [7]. They did not, however, provide any derivation or connection to estimation theory for their formulas.

This form of measurement processing is an alternative to *sequential processing* of scalar measurements [17] which avoids the need for recursively updating the estimate covariance for each new measurement.

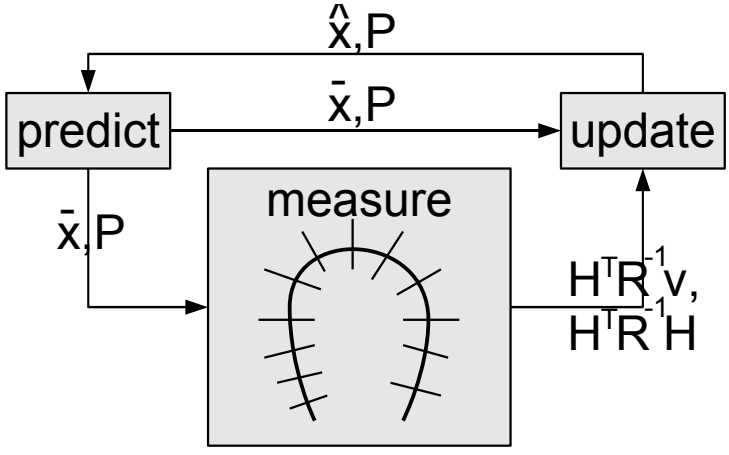


Figure 7.5: Overview over sequential contour tracking with measurement innovations in information space.

Chapter 8

Left Ventricle Model

This thesis uses a truncated ellipsoid contour as model for the left ventricle. It is inspired by the truncated ellipsoid model presented by Park in [21], although the degrees of freedoms has been somewhat limited to regularize the model.

8.1 Contour Template

The set of points defined by the circle equation, with z truncated to values below one using the truncation parameter T:

$$\begin{aligned} x^2 + y^2 + z^2 = 1, &\quad \text{for } z \in [-1, \sin(T)],\ T \leq \pi/2 \\ x^2 + y^2 \leq 1 - \sin^2(T), &\quad \text{for } z = \sin(T) \end{aligned}$$

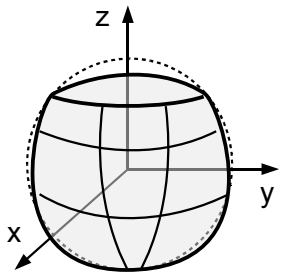


Figure 8.1: Truncated spherical contour

8.1.1 Parametric form

The ellipsoid equation can also be expressed in parametric form, using parameters u and v, for apical-basal movement and short-axis rotation respectively. The value of u is limited to

values lower than $\pi/2$. The main contour will then satisfy the following equation:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \cos(u)\cos(v) \\ \cos(u)\sin(v) \\ \sin(u) \end{bmatrix} \qquad \begin{matrix} T \leq \pi/2 \\ u \in [-\pi/2,\, T] \\ v \in [0, 2\pi) \end{matrix}$$

and for the truncation plane

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f\ \cos(T)\cos(v) \\ f\ \cos(T)\sin(v) \\ \sin(T) \end{bmatrix} \qquad \begin{matrix} f \in [0,\, 1] \\ v \in [0, 2\pi) \end{matrix}$$

A nice property of this contour model is that all contour normals, with the exception of the truncation plane, are parallel to the associated point coordinate. A separate model for contour normals is therefore not needed.

8.1.2 Contour Volume

The total contour volume can be calculated using integration. The calculations are simplified by dividing the spherical volume into two subvolumes, V_1 and V_2, where V_1 is an ordinary half-sphere ($z \in [-1,\, 0)$) and V_2 is a truncated half-sphere ($z \in [0,\, T]$).

For V_1 we get the following volume using standard geometry:

$$V_1 = 2\pi/3$$

V_2 must, however, be derived using integration:

$$\begin{aligned} A(z) &= \pi(1 - z^2) \\ V_2 &= \int_0^T A(z)dz \\ V_2 &= \int_0^T \pi(1 - z^2)dz \\ V_2 &= \pi[z - z^3/3]_0^T \\ V_2 &= \pi(T - T^3/3) \end{aligned}$$

the total contour volume thus becomes:

$$\begin{aligned} V &= V_1 + V_2 \\ V &= 2\pi/3 + \pi(T - T^3/3) \end{aligned}$$

The expression above is the volume of the *initial contour*. Subsequent contour deformations will typically alter this volume, particularly if scaling is present. But deformation models can none the less be constructed in such a way that most parameters are volume-preserving, while others affect the volume of the resulting contour in a controlled manner.

8.2 Deformation Modes

Pursuing the approach by Park et al. in [21] we define a deformation model for the truncated sphere consisting of the following parameters:

- Translation (t_x, t_y, t_z).
- Scaling (s_x, s_y, s_z).
- Rotation/orientation (r_x, r_y).
- "Bending" (c_x, c_y).

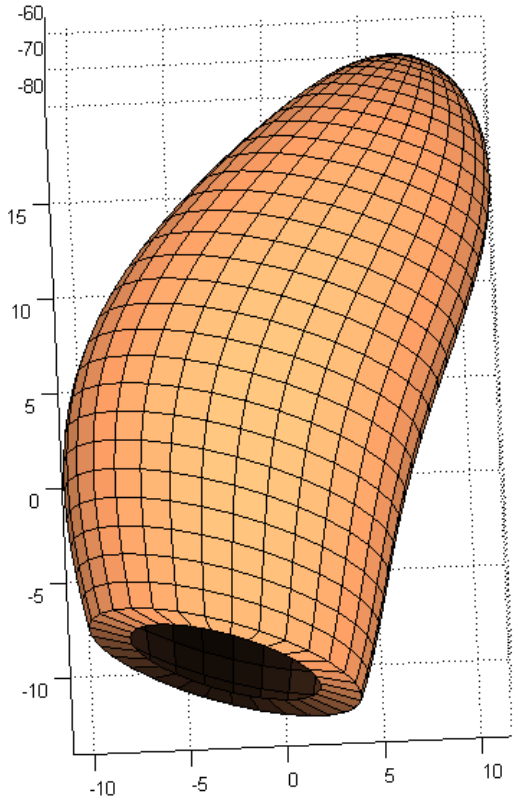


Figure 8.2: Example illustration of a deformed ellipsoid contour.

In total, this yields a state vector containing 10 parameters:

$$\mathbf{x} = \begin{bmatrix} t_x & t_y & t_z & s_x & s_y & s_z & r_x & r_y & c_x & c_y \end{bmatrix}^T$$

The proposed deformation model is then defined to be as follows:

$$\mathbf{p} = \mathcal{D}(\mathbf{p}_0, \mathbf{x})$$

$$\mathbf{p} = \mathcal{R}ot_x(\mathbf{x})\mathcal{R}ot_y(\mathbf{x})\mathcal{S}cale(\mathbf{x})\mathcal{B}end(\mathbf{x}, \mathbf{p}_0) + \mathcal{T}rans(\mathbf{x})$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(r_x) & \sin(r_x) \\ 0 & -\sin(r_x) & \cos(r_x) \end{bmatrix} \begin{bmatrix} \cos(r_y) & 0 & -\sin(r_y) \\ 0 & 1 & 0 \\ \sin(r_y) & 0 & \cos(r_y) \end{bmatrix} \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} \begin{bmatrix} x_0 + c_x \cos(\pi z_0) \\ y_0 + c_y \cos(\pi z_0) \\ z_0 \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

Based on the deformation model above, it is obvious that the volume of the deformed contour is invariant to all deformations except scaling. The volume is also linear in each of the scaling axes. The resulting contour volume hence becomes:

$$V = |s_x\, s_y\, s_z\, \left(2\pi/3 + \pi(T - T^3/3)\right)|$$

8.2.1 Spatial Jacobian

The deformation parameters are here considered constants. Ignoring rotations, we get the following Jacobian matrix:

$$\mathbf{J}_{\mathbf{p}}\mathcal{D}(\mathbf{p}_0, \mathbf{x}) = \begin{bmatrix} s_x & 0 & -\pi s_x c_x \sin(\pi z_0) \\ 0 & s_y & -\pi s_y c_y \sin(\pi z_0) \\ 0 & 0 & s_z \end{bmatrix}_{\mathbf{p}=\mathbf{p}_0}$$

WARNING: The Jacobian above is a simplification that does not account for rotations! The actual Jacobian matrix is found using a computer algebra system, and does unfortunately not look as elegant.

8.2.2 State-space Jacobian

The input coordinates are here considered constants. Ignoring rotations, we get the following Jacobian matrix:

$$\mathbf{J}_{\mathbf{x}}\mathcal{D}(\mathbf{p}_0, \mathbf{x}) = \begin{bmatrix} 1 & 0 & 0 & x_0 + c_x\cos(\pi z_0) & 0 & 0 & s_x\cos(\pi z_0) & 0 \\ 0 & 1 & 0 & 0 & y_0 + c_y\cos(\pi z_0) & 0 & 0 & s_Y\cos(\pi z_0) \\ 0 & 0 & 1 & 0 & 0 & z_0 & 0 & 0 \end{bmatrix}$$

WARNING: The Jacobian above is a simplification that does not account for rotations! The actual Jacobian matrix is found using a computer algebra system, and does unfortunately not look as elegant.

8.3 Future Extensions

This model can later be extended to become a full volumetric model which also specifies wall thickness. This would make it possible to use adaptive edge models coupled to predicted wall thickness along the contour. It is also possible to extend the model to contain more degrees of freedom for local deformation, which can be based on clinical analysis of tagged MR images, as done by Remme et al. in [24] and Luo et al. in [18].

Chapter 9

Kinematic Model

Kinematic models are used to predict contour state between successive frames. Such models act as *a priori* knowledge, yielding both a prediction for the state vector, and a covariance matrix, specifying prediction uncertainty. The prediction can then be used as a starting point for more accurate refinements, called updates, where the prediction is combined with measurements from the current frame to form more accurate estimates.

9.1 Kinematic Model for Contour Tracking

Most types of video tacking, including contour tracking, deals with moving, deformable objects that are nonstationary both in shape, alignment and position. This adds to the complexity of state prediction, since simple state estimates from the previous frame does not suffice as input for a kinematic model. This is because contour state vectors lack any concept of motion, or rate of change. A strategy for capturing temporal development in addition to the spatial is therefore required.

One solution to this problem is to augment the state vector to also contain rate-of-change values for all of the original parameters. A kinematic model can then be derived, based on the discretization of a continuous position-velocity model, as was originally done by Blake et al. in [6]. A simpler, more intuitive approach is simply to augment the state vector to contain the last two successive state estimates, as was later done in [8]. This both alleviates the discretization problem, and makes model interpretation more intuitive. The latter approach is therefore pursued in this thesis.

The desire to estimate motion in addition to position therefore leads to a kinematic model based on the last two state estimates:

$$\mathbf{x}_{k+1} = \mathbf{A}_1\mathbf{x}_k + \mathbf{A}_2\mathbf{x}_{k-1} + \mathbf{B}_0\mathbf{w}_k$$

Augmenting the state vector then brings the model on standard form

$$\begin{bmatrix} \mathbf{x}_k \\ \mathbf{x}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{A}_2 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_{k-1} \\ \mathbf{x}_k \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_0 \end{bmatrix} \mathbf{w}_k$$

Measurements are, however, only based on the current state. Any measurement model does

therefore require a slight change in form to reflect this:

$$z_k = \mathbf{H}\mathbf{x}_k + v_k$$
$$z_k = \mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\begin{bmatrix}\mathbf{x}_{k-1} \\ \mathbf{x}_k\end{bmatrix} + r\, v_k$$

When put on standard form, the total model becomes as follows:

$$\mathbf{X}_{k+1} = \mathbf{F}\mathbf{X}_k + \mathbf{B}^a\mathbf{w}_k$$
$$\mathbf{z}_k = \mathcal{H}\mathbf{X}_k + r\,\mathbf{v}_k$$

where

$$\mathbf{X}_k \equiv \begin{bmatrix}\mathbf{x}_{k-1} \\ \mathbf{x}_k\end{bmatrix},\ \mathbf{F} = \begin{bmatrix}\mathbf{0} & \mathbf{I} \\ \mathbf{A}_2 & \mathbf{A}_1\end{bmatrix},\ \mathbf{B}^a = \begin{bmatrix}\mathbf{0} \\ \mathbf{B}_0\end{bmatrix},\ \mathcal{H} = \mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}$$

leading to covariance matrices on the following form

$$\mathbf{P} = \begin{bmatrix}\mathbf{P}_2 & \mathbf{P}_1^T \\ \mathbf{P}_1 & \mathbf{P}_0\end{bmatrix},\ \mathbf{Q} = \begin{bmatrix}\mathbf{0} \\ \mathbf{B}_0\end{bmatrix}\begin{bmatrix}\mathbf{0} & \mathbf{B}_0^T\end{bmatrix},\ \mathbf{R} = \begin{bmatrix}r\end{bmatrix}$$

9.2 Interpretation of Model Coefficients

The model coefficients are most easily understood by considering the three main types of models encountered, namely:

Inertial - Rate of change is here constant, leading to undamped constant velocity motion. The kinematic model is inertial if $\mathbf{A}_1 = 2\mathbf{I}$ and $\mathbf{A}_2 = -\mathbf{I}$. The model will then assume contours continuing at undisturbed velocity.

Damped - Rate of change is here strongly decreasing, leading to sudden stops in all motion. The kinematic model is totally damped if $\mathbf{A}_1 = 1\mathbf{I}$ and $\mathbf{A}_2 = \mathbf{0}$. All motion is here assumed to suspend instantly within the next frame.

Regularized - Contour state is here pulled towards a prior mean state, which will regularize the state to prevent unconstrained state and motion. The kinematic model is regularized if $\mathbf{A}_1 + \mathbf{A}_2 < \mathbf{I}$.

By combining the three classes mentioned above, one can construct intermediate models, with properties somewhere in between the extremes. Introduction of parameter $dam \in [0, 1]$, which controls damping strength, and $reg \in [0, 1]$, which controls regularization strength, yields:

$$\mathbf{A}_1 = (2 - dam)\, reg\, \mathbf{I}$$
$$\mathbf{A}_2 = (dam - 1)\, reg\, \mathbf{I}$$

Model construction this way, through *dam* and *reg*, is both simple and intuitive. It also opens up for *subspace models*, where different state vector elements have different kinematic properties. To illustrate this, one would typically desire to dampen object rotation much more than translations, since rotations usually ceases faster than translative motion. Regularizations would, however, typically be stronger on translation and scaling, than on rotations, since there often exist no notion of "correct" alignment for the object being tracked. These modeling choices are, however, very domain dependent, and might vary considerable from application to application

Chapter 10

Kalman Filter Equations for Tracking

The special problem structure encountered in contour tracking introduces several challenges when it comes to the Kalman filter implementation. The numbers of measurements are often an order of magnitude greater than the state dimension, which leads to inversion of unnecessarily large matrices when performing measurement updates. The special process structure, with augmented state vectors containing the last two successive state estimates, also opens up for more efficient implementations compared to the classical Kalman filter.

The goal for this chapter is thus to derive a variant of the Kalman filter capable of handling the challenges mentioned above in an efficient manner. To summarize, we desire:

- Measurements on *information filter* form, where all measurement information is efficiently summed into information-vectors and -matrices, with size invariant to the number of measurements, to enable efficient measurement processing.
- Exploit the model structure to minimize the number of matrix inversion, as well as the size of the matrices to be inverted. A major challenge here is to prevent inversion of any of the large error covariance matrices associated with the augmented state vectors.

The conceptual technique was first presented by Blake et al. in [6] and [8], but these publications were using a more inefficient *sequential processing* approach to Kalman filtering, where the posterior error covariance matrix has to be updated once for every singe measurement taken, which in turn leads to a more inefficient implementation.

The derivations in this chapter do instead culminate in what was originally presented by Blake and Isard in *Active Contours* [7]. Blake et al. presented the Kalman filter based algorithm there, as well as an alternative formulation which is not pursued in this thesis, but provided neither proof nor derivation for the results. This chapter does therefore attempt to bridge that gap by providing a full derivation of the tracking equations. A further simplification compared to [7] is also derived for the posterior error covariance equations, leading to an even more efficient implementation.

10.1 State Prediction Equations

State prediction is performed directly, using the kinematic model. No further optimization is possible here.

$$\bar{\mathbf{X}}_{k+1} = \mathbf{F}\hat{\mathbf{X}}_k$$
$$\begin{bmatrix} \bar{\mathbf{x}}_k \\ \bar{\mathbf{x}}_{k+1} \end{bmatrix} = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{A}_2 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{x}}_{k-1} \\ \hat{\mathbf{x}}_k \end{bmatrix}$$

The error covariance for the prediction can, however, be expanded by subdividing the covariance into four submatrices. A modest simplification is then possible:

$$\bar{\mathbf{P}} = \mathbf{F}\hat{\mathbf{P}}\mathbf{F}^T + \mathbf{Q}$$
$$\begin{aligned}
\begin{bmatrix} \bar{\mathbf{P}}_2 & \bar{\mathbf{P}}_1^T \\ \bar{\mathbf{P}}_1 & \bar{\mathbf{P}}_0 \end{bmatrix} &= \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{A}_2 & \mathbf{A}_1 \end{bmatrix} \begin{bmatrix} \hat{\mathbf{P}}_2 & \hat{\mathbf{P}}_1^T \\ \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{A}_2^T \\ \mathbf{I} & \mathbf{A}_1^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} \\ \mathbf{B}_0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{B}_0^T \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_0 \\ \mathbf{A}_2\hat{\mathbf{P}}_2 + \mathbf{A}_1\hat{\mathbf{P}}_1 & \mathbf{A}_2\hat{\mathbf{P}}_1^T + \mathbf{A}_1\hat{\mathbf{P}}_0 \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{A}_2^T \\ \mathbf{I} & \mathbf{A}_1^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0\mathbf{B}_0^T \end{bmatrix} \\
&= \begin{bmatrix} \hat{\mathbf{P}}_0 & \hat{\mathbf{P}}_1\mathbf{A}_2^T + \hat{\mathbf{P}}_0\mathbf{A}_1^T \\ \mathbf{A}_2\hat{\mathbf{P}}_1^T + \mathbf{A}_1\hat{\mathbf{P}}_0 & (\mathbf{A}_2\hat{\mathbf{P}}_2 + \mathbf{A}_1\hat{\mathbf{P}}_1)\mathbf{A}_2^T + (\mathbf{A}_2\hat{\mathbf{P}}_1^T + \mathbf{A}_1\hat{\mathbf{P}}_0)\mathbf{A}_1^T \end{bmatrix} + \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{B}_0\mathbf{B}_0^T \end{bmatrix}
\end{aligned}$$

10.1.1 State Prediction Implementation

The derivations above yield a state prediction implementation on the following form, when we subdivide and expand the matrix expression for error covariance:

$$\begin{aligned}
\bar{\mathbf{x}}_{k+1} &= \mathbf{A}_1\hat{\mathbf{x}}_k + \mathbf{A}_2\hat{\mathbf{x}}_{k-1} \\
\bar{\mathbf{P}}_2 &= \hat{\mathbf{P}}_0 \\
\bar{\mathbf{P}}_1 &= \mathbf{A}_2\hat{\mathbf{P}}_1^T + \mathbf{A}_1\hat{\mathbf{P}}_0 \\
\bar{\mathbf{P}}_0 &= (\mathbf{A}_2\hat{\mathbf{P}}_2 + \mathbf{A}_1\hat{\mathbf{P}}_1)\mathbf{A}_2^T + (\mathbf{A}_2\hat{\mathbf{P}}\mathbf{1}^T + \mathbf{A}_1\hat{\mathbf{P}}_0)\mathbf{A}_1^T + \mathbf{B}_0\mathbf{B}_0^T
\end{aligned}$$

10.2 Measurement Update Equations

As mentioned before, the augmented state formulation leads to a measurement model on the following form:

$$\begin{aligned}
z_i &= \mathcal{H}\mathbf{X} + v_i \\
&= \mathbf{H}\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix}\begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_0 \end{bmatrix} + v_i
\end{aligned}$$

The measurement update step can be altered, by utilizing that the Kalman gain $\mathbf{K}_k \equiv \hat{\mathbf{P}}_k\mathcal{H}^T\mathbf{R}^{-1}$, and reformulating to account for measurements on information filter form:

$$\begin{aligned}
\hat{\mathbf{X}}_k &= \bar{\mathbf{X}}_k + \mathbf{K}_k v_k \\
\hat{\mathbf{X}}_k &= \bar{\mathbf{X}}_k + \hat{\mathbf{P}}_k\mathcal{H}^T\mathbf{R}^{-1}v_k \\
\hat{\mathbf{X}}_k &= \bar{\mathbf{X}}_k + \hat{\mathbf{P}}_k(\mathbf{H}\begin{bmatrix} \mathbf{0} & \mathbf{I} \end{bmatrix})^T\mathbf{R}^{-1}v_k \\
\hat{\mathbf{X}}_k &= \bar{\mathbf{X}}_k + \hat{\mathbf{P}}_k\begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}(\mathbf{H}^T\mathbf{R}^{-1}v_k)
\end{aligned}$$

where measurement innovations are efficiently summed into a measurement vector with dimension equal to the state dimension, using equation $\mathbf{H}^T\mathbf{R}^{-1}v_k$.

Update of the error covariance equations poses a different challenge. The great number of measurements compared to the state dimension leads to inversion of unnecessary large matrices, since the combined measurement error covariance matrix would have dimensions equal to the number of measurements. We therefore desire to use the information filter formulation of the Kalman filter, where measurement uncertainty are first summed into information matrices in information space, using the information filter formulation of the Kalman filter. This strategy leads to the following derivations:

$$\begin{aligned}
\hat{\mathbf{P}}^{-1} &= \bar{\mathbf{P}}^{-1} + \mathcal{H}^T\mathbf{R}^{-1}\mathcal{H} \\
\hat{\mathbf{P}}^{-1} &= \bar{\mathbf{P}}^{-1} + \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix} \\
\begin{bmatrix}\hat{\mathbf{P}}_2 & \hat{\mathbf{P}}_1^T\\ \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_0\end{bmatrix}^{-1} &= \begin{bmatrix}\bar{\mathbf{P}}_2 & \bar{\mathbf{P}}_1^T\\ \bar{\mathbf{P}}_1 & \bar{\mathbf{P}}_0\end{bmatrix}^{-1} + \begin{bmatrix}\mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\end{bmatrix}
\end{aligned}$$

This form, however, still requires inversion of matrices with dimensions equal to twice the state dimension ($2N_x$x$2N_x$), due to the augmented state vector. This problem can fortunately be avoided by a clever reformulation of the original Kalman filter, rejecting the information filter formulation just derived.

We instead focus on the standard Kalman equations, and try to adapt them to our needs. Insertion of our measurement model onto thee Kalman gain equation gives us:

$$\begin{aligned}
\mathbf{K} &= \bar{\mathbf{P}}\mathcal{H}^T(\mathcal{H}\bar{\mathbf{P}}\mathcal{H}^T + \mathbf{R})^{-1} \\
&= \bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T(\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T + \mathbf{R})^{-1}
\end{aligned}$$

By redefining the Kalman gain to include $(\mathbf{H}^T\mathbf{R}^{-1})^{-1}$ one can modify the gain equation above to efficiently accept measurement covariance in information space:

$$\begin{aligned}
\mathcal{K} &\equiv \mathbf{K}(\mathbf{H}^T\mathbf{R}^{-1})^{-1} \\
&= \bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T(\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T + \mathbf{R})^{-1}(\mathbf{H}^T\mathbf{R}^{-1})^{-1} \\
&= \bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{R})^{-1} \\
&= \bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}\mathbf{H}^T(\mathbf{H}^T)^{-1} + \mathbf{H}^T\mathbf{R}^{-1}\mathbf{R}(\mathbf{H}^T)^{-1})^{-1} \\
&= \bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix}(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}\bar{\mathbf{P}}\begin{bmatrix}\mathbf{0}\\ \mathbf{I}\end{bmatrix} + \mathbf{I})^{-1} \\
\begin{bmatrix}\mathcal{K}_1\\ \mathcal{K}_0\end{bmatrix} &= \begin{bmatrix}\bar{\mathbf{P}}_1^T\\ \bar{\mathbf{P}}_0\end{bmatrix}(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\bar{\mathbf{P}}_0 + \mathbf{I})^{-1}
\end{aligned}$$

where the modified Kalman gain is decomposed into two blocks.

With the alternative Kalman gain equation in hand, we can derive a new equation for the updated error covariance for the estimate:

$$\begin{aligned}
\hat{\mathbf{P}} &= (\mathbf{I} - \mathbf{K}\mathcal{H})\bar{\mathbf{P}} \\
&= (\mathbf{I} - \mathbf{K}\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix})\bar{\mathbf{P}} \\
&= (\mathbf{I} - \mathbf{K}(\mathbf{H}^T\mathbf{R}^{-1})^{-1}(\mathbf{H}^T\mathbf{R}^{-1})\mathbf{H}\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix})\bar{\mathbf{P}} \\
&= (\mathbf{I} - \mathcal{K}(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix})\bar{\mathbf{P}} \\
\begin{bmatrix}\hat{\mathbf{P}}_2 & \hat{\mathbf{P}}_1^T \\ \hat{\mathbf{P}}_1 & \hat{\mathbf{P}}_0\end{bmatrix} &= \begin{bmatrix}\bar{\mathbf{P}}_2 & \bar{\mathbf{P}}_1^T \\ \bar{\mathbf{P}}_1 & \bar{\mathbf{P}}_0\end{bmatrix} - \begin{bmatrix}\mathcal{K}_1 \\ \mathcal{K}_0\end{bmatrix}(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\begin{bmatrix}\bar{\mathbf{P}}_1 & \bar{\mathbf{P}}_0\end{bmatrix}
\end{aligned}$$

Then, by looking back at the estimate update equation, we can easily see that the modified Kalman gain is simply a submatrix of the updated error covariance:

$$\begin{aligned}
\mathcal{K} &= \mathbf{K}(\mathbf{H}^T\mathbf{R}^{-1})^{-1} \\
&= \hat{\mathbf{P}}_k\begin{bmatrix}\mathbf{0} \\ \mathbf{I}\end{bmatrix}\mathbf{H}^T\mathbf{R}^{-1}(\mathbf{H}^T\mathbf{R}^{-1})^{-1} \\
&= \hat{\mathbf{P}}_k\begin{bmatrix}\mathbf{0} \\ \mathbf{I}\end{bmatrix} \\
\begin{bmatrix}\mathcal{K}_1 \\ \mathcal{K}_0\end{bmatrix} &= \begin{bmatrix}\hat{\mathbf{P}}_1^T \\ \hat{\mathbf{P}}_0\end{bmatrix}
\end{aligned}$$

This relationship has to the best of my knowledge not previously been revealed, nor exploited in the literature.

The alternative Kalman gain definition also fits nicely into the measurement update, which then elegantly becomes:

$$\begin{aligned}
\hat{\mathbf{X}}_k &= \bar{\mathbf{X}}_k + \mathcal{K}(\mathbf{H}^T\mathbf{R}^{-1}v_k) \\
\begin{bmatrix}\hat{\mathbf{x}}_1 \\ \hat{\mathbf{x}}_0\end{bmatrix} &= \begin{bmatrix}\bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_0\end{bmatrix} + \begin{bmatrix}\mathcal{K}_1 \\ \mathcal{K}_0\end{bmatrix}(\mathbf{H}^T\mathbf{R}^{-1}v_k)
\end{aligned}$$

10.2.1 Measurement Update Implementation

The derivations above yields a measurement update implementation on the following form, when we decompose all augmented matrices into blocks which can be evaluated individually, and expand all expressions:

By expanding and subdividing the modified Kalman gain equations, we get:

$$\begin{aligned}
\mathcal{K}_1 &= \bar{\mathbf{P}}_1^T(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\bar{\mathbf{P}}_0 + \mathbf{I})^{-1} \\
\mathcal{K}_0 &= \bar{\mathbf{P}}_0(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H}\bar{\mathbf{P}}_0 + \mathbf{I})^{-1}
\end{aligned}$$

Similarly, the measurement update becomes as follows:

$$\begin{aligned}
\hat{\mathbf{x}}_1 &= \bar{\mathbf{x}}_1 + \mathcal{K}_1(\mathbf{H}^T\mathbf{R}^{-1}v_k) \\
\hat{\mathbf{x}}_0 &= \bar{\mathbf{x}}_0 + \mathcal{K}_0(\mathbf{H}^T\mathbf{R}^{-1}v_k)
\end{aligned}$$

and by exploiting the relationship between Kalman gain and posterior error covariance derived above, we get the following posterior covariance implementation:

$$\begin{aligned}\hat{\mathbf{P}}_2 &= \bar{\mathbf{P}}_2 - \mathcal{K}_1(\mathbf{H}^T\mathbf{R}^{-1}\mathbf{H})\bar{\mathbf{P}}_1 \\ \hat{\mathbf{P}}_1 &= \mathcal{K}_1^T \\ \hat{\mathbf{P}}_0 &= \mathcal{K}_0\end{aligned}$$

10.3 Measurement Likelihood

Measurements are modeled as stochastic processes, affected by multivariate Gaussian noise. The probability density distribution for singe measurements hence becomes a scalar Gaussian probability density function (pdf) centered around the predicted position, with a variance equal to the measurement noise.

$$z_i \sim \mathcal{N}(\bar{z}, s) = \mathcal{N}(\mathbf{h}_i^T\mathbf{x}_i,\, \mathbf{h}_i^T\bar{\mathbf{P}}\mathbf{h}_i + r)$$

This leads to the following measurement likelihood function:

$$p(z_i|\mathbf{x}_i) = \frac{1}{(2\pi)^{1/2}\left|\mathbf{h}_i^T\bar{\mathbf{P}}\mathbf{h}_i + r\right|^{1/2}} \exp\left(-\frac{1}{2}(z_i - \mathbf{h}_i\mathbf{x}_i)^T(\mathbf{h}_i^T\bar{\mathbf{P}}\mathbf{h}_i + r)^{-1}(z_i - \mathbf{h}_i\mathbf{x}_i)\right)$$

10.3.1 Validation Gate

Knowledge of the likelihood associated with the measurements can be used to filter out improbable measurements, which can help improve tracking robustness.

The *de facto* way of performing this is by using the measurement innovation and error covariance to calculate the *Normalized Innovation Squared* (NIS) as defined by Bar-Shalom et al. in [4]. With scalar measurements, the NIS becomes:

$$NIS_i \propto v_i^T(\mathbf{h}_i^T\bar{\mathbf{P}}\mathbf{h}_i + r)^{-1}v_i$$

We clearly see that NIS grows proportional to the square of the innovations. Large measurement innovations will therefore typically lead to large NIS values, which can lead to the measurements being rejected.

Chapter 11

Edge Detection

Edge detection is the process of detecting the presence and position of edges in images. Edges are usually defined to be any significant change in image intensity occurring over a short spatial scale. The strategies for detecting edges are many and diverse. The most commonly used procedure is probably spatial derivative-filtering, a process that enhances any changes in image intensity. Subsequent thresholding will then reveal the position of any edges present. A problem with this approach is that it not only enhances edges, but also noise, which limits the robustness of the method. Applications needing robust edge detection will therefore often require usage of different approaches.

The normal approach to edge detection is usually to process the entire image, something that can be computationally demanding since images usually are two- dimensional, consisting of many hundred thousand, or millions of pixels. The contour-tracking approach does, however, improve this situation, since only contour normals are examined. Processing of the entire image is thus superfluous, opening up for simpler one-dimensional edge detection, where only a few pixels have to be examined for each edge.

This thesis presents two different edge detection algorithms, which are both based on statistical intensity models.

11.1 Step Edge Model

Robust edge detection is paramount to the success of any contour-tracking application, because tracking performance is always limited by measurement quality. The construction of a robust edge-detector must therefore be done with great care, so that it minimizes the chance of detecting spurious edges in noisy areas. An edge-model that exhibits robust characteristics is the *step model* [23] for edges. This model models edges as a transition in image intensity, for one plateau to another.

The advantage of this model is that it enables simple calculation of model mismatch, which we define to be the *sum of squared errors (SSE)* between the model and the data. This criterion can be calculated efficiently, as shown below. Edge candidates with large model mismatch can also safely be ignored as false alarms, resulting from image noise or clutter.

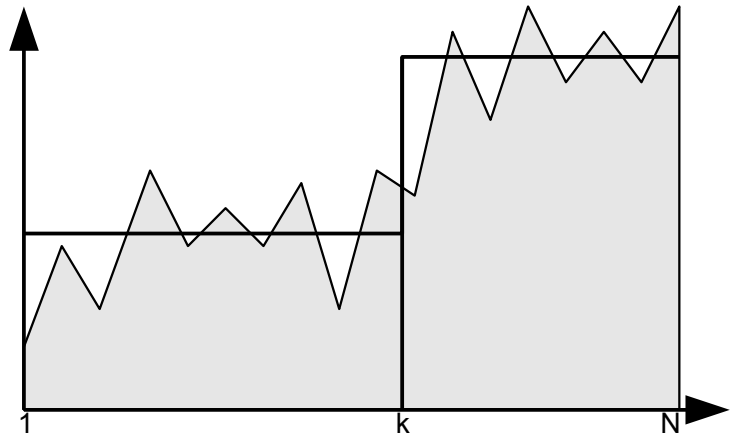


Figure 11.1: Step model for edges

The model mismatch criteria listed above leads us to the definition of *variance* in statistics, which can be shown to equal [25]:

$$var(x) = E\{(x - E\{x\})^2\} = E\{x^2\} - E\{x\}^2$$

The sum of squared errors (SSE) for a normal-line consisting of N pixels (x_1 through x_N) is thus:

$$\begin{aligned} SSE &= N\ var(x) \\ &= N\left[E\{x^2\} - E\{x\}^2\right] \\ &= N\left[\frac{1}{N}\sum_{i=1}^{N} x_i^2 - \left(\frac{1}{N}\sum_{i=1}^{N} x_i\right)^2\right] \\ &= \sum x^2 - \frac{1}{N}(\sum x)^2 \end{aligned}$$

11.1.1 Edge Detection Algorithm

In the step-edge case, we must introduce variable k, which identifies the edge position. SSE calculation is then split up into two parts, before and after the edge-position:

$$\begin{aligned} SSE(k) &= k\left[\frac{1}{k}\sum_{i=1}^{k} x_i^2 - \left(\frac{1}{k}\sum_{i=1}^{k} x_i\right)^2\right] + (N-k)\left[\frac{1}{N-k}\sum_{i=k+1}^{N} x_i^2 - \left(\frac{1}{N-k}\sum_{i=k+1}^{N} x_i\right)^2\right] \\ &= \sum_{i=1}^{N} x_i^2 - \frac{1}{k}\left(\sum_{i=1}^{k} x_i\right)^2 - \frac{1}{N-k}\left(\sum_{i=k+1}^{N} x_i\right)^2 \\ &= \sum x^2 - \frac{1}{k}cum[k]^2 - \frac{1}{N-k}icum[k]^2 \end{aligned}$$

where $cum[k]$ and $icum[k]$ are the forward and reverse cumulative sums of the plot, respectively. The most probable edge will then be the value of k that minimizes SSE. This problem can be solved in linear time by first calculating the forward and reverse cumulative sums, and then testing for which value of k that minimizes the expression.

11.2 Peak Edge Model

An alternative to the step edge model is the *peak model* [1] for edges. This model used two intermediate transitions in intensity to model an edge. The distance between the transitions are assumed constant and does typically lie in the interval (20% - 40% of N).

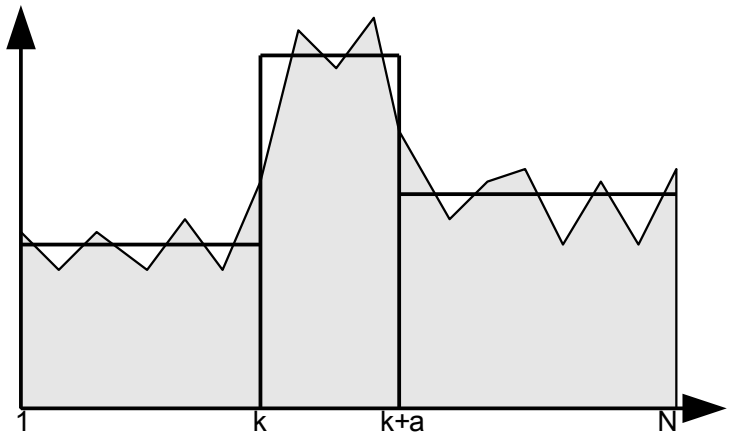


Figure 11.2: Peak model for edges

Following the same steps as above, this yield the following equations:

$$
\begin{aligned}
SSE(k) &= k\left[\frac{1}{k}\sum_{i=1}^{k}x_i^2-\left(\frac{1}{k}\sum_{i=1}^{k}x_i\right)^2\right]+A\left[\frac{1}{A}\sum_{i=k+1}^{k+A}x_i^2-\left(\frac{1}{A}\sum_{i=k+1}^{k+A}x_i\right)^2\right] \\
&\quad+(N-k-A)\left[\frac{1}{N-k-A}\sum_{i=k+A+1}^{N}x_i^2-\left(\frac{1}{N-k-A}\sum_{i=k+A+1}^{N}x_i\right)^2\right] \\
SSE(k) &= \sum_{i=1}^{N}x_i^2-\frac{1}{k}\left(\sum_{i=1}^{k}x_i\right)^2-\frac{1}{A}\left(\sum_{i=k+1}^{k+A}x_i\right)^2-\frac{1}{N-k-A}\left(\sum_{i=k+A+1}^{N}x_i\right)^2 \\
&= \sum x^2-\frac{1}{k}cum[k]^2-\frac{1}{A}wcum_A[k]^2-\frac{1}{N-k-A}icum[k+A]^2
\end{aligned}
$$

where $wcum[k]$ is a windowed cumulative sums of x_i from k to $k+A$. $cum[k]$ and $icum[k]$ are equal to their counterparts in the step-model. The most probable edge will then be the value of k that minimizes SSE. This problem can be solved in linear time if we start by assuming a fixed value of A. The most probable edge position is found, just like above, by testing for which value of k that minimizes the expression.

[1] The author has been unable to discover any references on prior usage of this model. It is, however, believed that prior art does exists. The model should therefore not be considered as original research.

Part III

Results

Chapter 12

Automatic Tracking Experiment

The aim of the automatic tracking experiment is to *validate the feasibility* of the tracking method proposed. The validation is performed by testing the library's ability to automatically track the left ventricle (LV) in 3D echocardiography recordings, entirely without any user input.

A subjective metric is used to evaluate the accuracy of the method, were the tracking results for each recording is categorized as either *good*, *adequate*, *fair* and *poor* based on apparent correspondence between the LV-contour found in the recordings and the contour estimated by the framework.

12.1 Scope

The experiments performed in this chapter focuses primarily on the following problems encountered in LV-tracking:

- Acquire lock on LV initially, without any user input.
- Track the dominant LV motion and deformations, including translation, orientation and to some extent scaling and bending.

Exact contour tracking and wall segmentation has not been the goal for these experiments. Accurate segmentation is a more difficult problem, which would probably require a more advanced LV model, and will instead be in focus for future research in this area.

12.2 Setup Configuration

The following setup configuration was used:

- Used the proposed deformable truncated ellipsoid model as tracking template, with deformation parameters for translation, rotation, scaling and bending in all three dimensions.
- Used a "step"-based edge model.
- 426 edge normals. Each of 20mm search length, with samples every millimeter.

- Used edge strength magnitude as validation gate, with a minimum intensity difference of 12 units.
- Default/prior contour placed in the center of the volume without any user intervention.

12.2.1 Configuration File Used

The following common configuration file was used for *almost* all of the tracking experiments:

```
<object particles="0">

<input type="beam3d" filename="../ClinicalData/hortenXX.raw3d"/>

<kinematic>
    <translate damp="0.8" reg="0.10" noise="10"/>
    <scale     damp="0.8" reg="0.20" noise="40"/>
    <rotate    damp="0.8" reg="0.10" noise="0.1"/>
    <deform    damp="0.8" reg="0.10" noise="0.1"/>
</kinematic>

<edge model="step" search_length="20" meas_noise="0.5">
    <color r="-7" g="-7" b="-7"/>
</edge>

<contour type="ellipsoid" deform="ventricle" periodic="true"
         x_center="0" y_center="0" z_center="-90">
          0  0   0
         15 20 -45
          0  0
          0  0
</contour>
</object>
```

The only exceptions to the common configuration file above were the following special cases in the training dataset:

- **Contrast recordings** - the edge model was inverted to account for higher intensity echo from blood in the cavity.
- **Parasternal views** - initial contour were rotated 60 degrees and translated 20mm upwards to approximate the real LV position and alignment.
- **4-chamber views** - initial position translated 10mm to the right to account for the LV not being in the center of the volume.
- Default/prior contour depth varied between 80mm and 90mm in the test-set (an undesired inconsistency made by me).

These anomalies were required since the tracking algorithm requires the initial contour to be in a fair proximity to the real ventricle contour.

12.3 Testing on the Training Dataset

A dataset consisting of 93 recordings, were 88 of them were present and in suitable shape for processing, were received from GE Vingmed Ultrasound. This dataset were not cleared for publication, so its usage was therefore restricted to train and tune the tracking framework. This section presents the best results achieved in this dataset, and serves to yield some insight to the tracking results achievable for the framework.

WARNING: This is the same dataset as was used to develop and tune the tracking algorithm. Results may therefore be better than what can be expected in a random dataset.

Results

Tracking experiments were performed a wide variety of cardiac 3D ultrasound recordings, with and without the use of contrast agents. The results are found in figures C.1, C.1 and C.3 in the appendix, with summaries in 12.1. Tracking were performed with *good* or *adequate* quality in 77% of the 88 recordings present in the dataset.

Screenshots showing the tracking performed in each of the recordings is provided as an appendix.
NOTICE: The tracking is performed in a complete 3D volume, even though visualization is limited to rendering a singe long-axis view through the volume.

Overall tracking results			
Quality	Count	Percentage	Comment
Good	52	59%	Automatic tracking of LV position and long-axis performed well.
Adequate	16	18%	Automatic tracking of LV position and long-axis performed with reduced accuracy.
Fair	14	16%	Automatic tracking of LV position and long-axis performed with low accuracy.
Poor	6	7%	Unable to automatically track of LV position and long-axis
n/a	5		Files missing or corrupt. Unable to perform tracking in them.

Table 12.1: Overall result statistic from automatic tracking within the GE Vingmed Ultrasound dataset.

12.4 Testing on Independent Validation Dataset

A collection of 21 3D-echocardiography recordings, where 50% of the patients had a heart illness, were received from Brage Amundsen at ISB/NTNU. Tracking in this dataset serves as independent validation of the method. The same configuration file as in the previous experiment were used, but here entirely without exceptions since no parasternal recordings, recordings showing all four heart chamber, or contrast recordings were present in this dataset.

Results

The results are found in figure 12.2 with summaries in 12.3. The recordings were of higher quality compared to the test set, which made tracking easier. Tracking were performed with *good* or *adequate* quality in 90% of the 21 recordings present in the dataset.

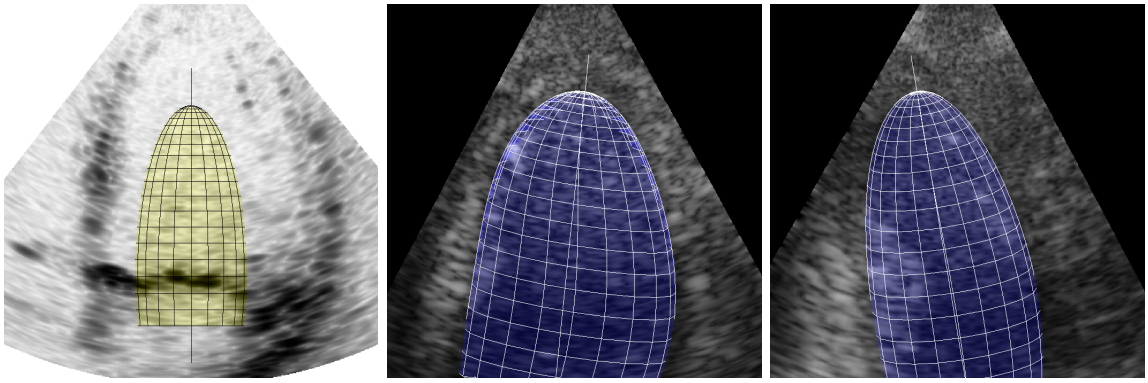
Screenshots showing the tracking performed in each of the recordings is provided at the end of this chapter.
NOTICE: The tracking is performed in a complete 3D volume, even though visualization is limited to rendering a singe long-axis view through the volume.

Detailed tracking results		
Recording no	Tracking result	Comments
Recording 1 (2006 01 13 1b)	good	Very poor contrast.
Recording 2 (2006 01 13 2a)	adequate	
Recording 3 (2006 01 13 3c)	adequate	A bit unstable near apex.
Recording 4 (2006 01 13 4b)	good	
Recording 5 (2006 01 20 1a)	good	.
Recording 6 (2006 01 20 2c)	good	Very poor contrast.
Recording 7 (2006 01 20 3d)	good	.
Recording 8 (2006 01 20 4b)	good	.
Recording 9 (2006 01 20 5b)	good	Apex partly outside volume.
Recording 10 (2006 02 10 1a)	good	.
Recording 11 (2006 02 10 2b)	good	.
Recording 12 (2006 02 10 3a)	adequate	Poor contrast.
Recording 13 (2006 02 10 4a)	good	.
Recording 14 (2006 02 10 5b)	good	.
Recording 15 (2006 02 22 1a)	good	.
Recording 16 (2006 02 22 2e)	poor	Unable to track mitral plane.
Recording 17 (2006 03 08 1d)	good	.
Recording 18 (2006 03 08 2a)	good	.
Recording 19 (2006 03 08 3f)	good	.
Recording 20 (2006 03 21 1b)	good	.
Recording 21 (2006 03 21 2c)	fair	Problems tracking apex.

Table 12.2: Results of automatic tracking within the independent validation dataset.

Overall tracking results			
Quality	Count	Percentage	Comment
Good	16	76%	Automatic tracking of LV position and long-axis performed well.
Adequate	3	14%	Automatic tracking of LV position and long-axis performed with reduced accuracy.
Fair	1	5%	Automatic tracking of LV position and long-axis performed with low accuracy.
Poor	1	5%	Unable to automatically track of LV position and long-axis
n/a	0		Files missing or corrupt. Unable to perform tracking in them.

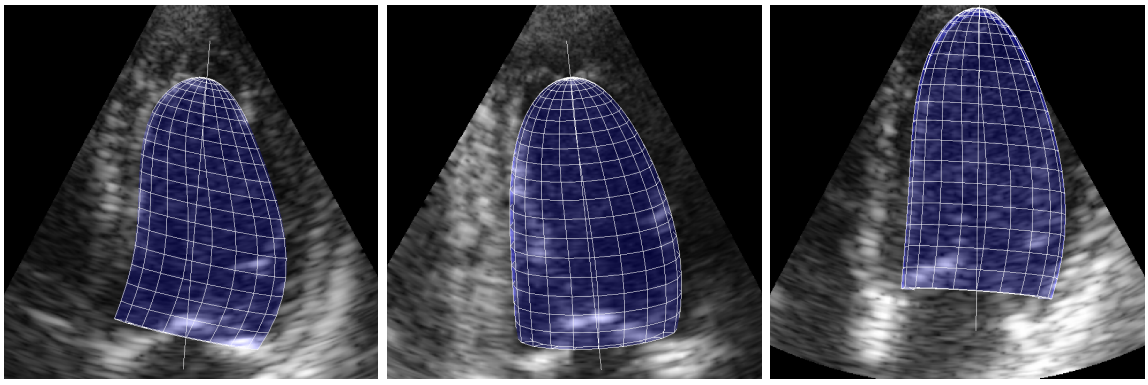
Table 12.3: Overall result statistic from automatic tracking within the independent validation dataset.



(a) Initial contour used in all recordings

(b) Recording 1 (azimuth)

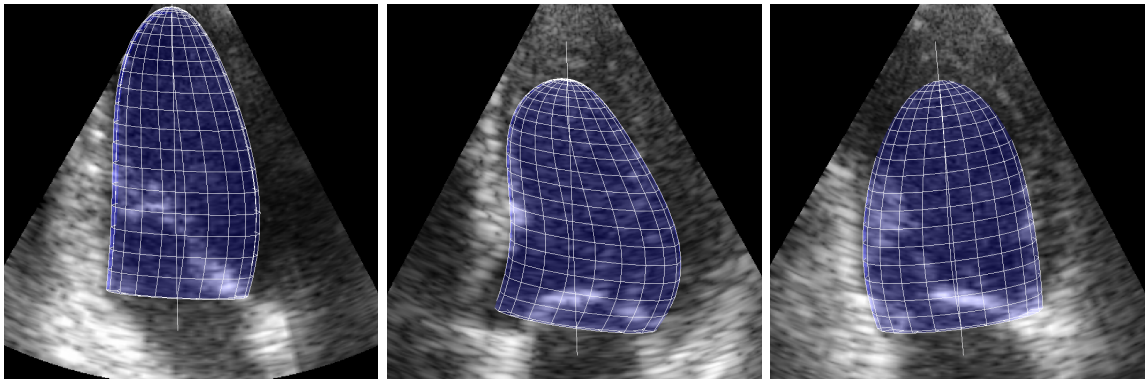
(c) Recording 1 (elevation)



(d) Recording 2 (azimuth)

(e) Recording 2 (elevation)

(f) Recording 3 (azimuth)

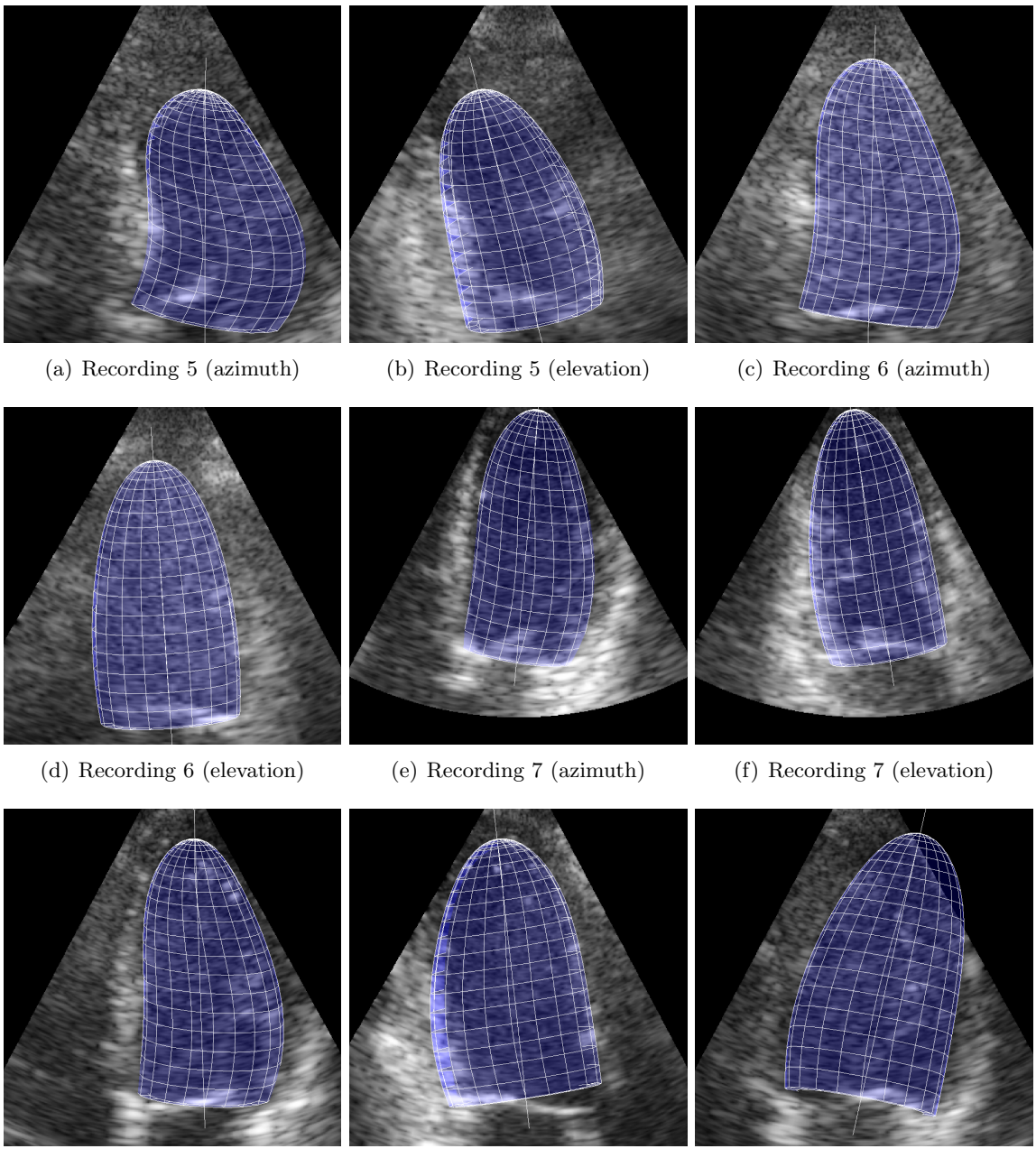


(g) Recording 3 (elevation)

(h) Recording 4 (azimuth)

(i) Recording 4 (elevation)

Figure 12.1: Screenshots from automatic tracking within the independent validation dataset (1/5).

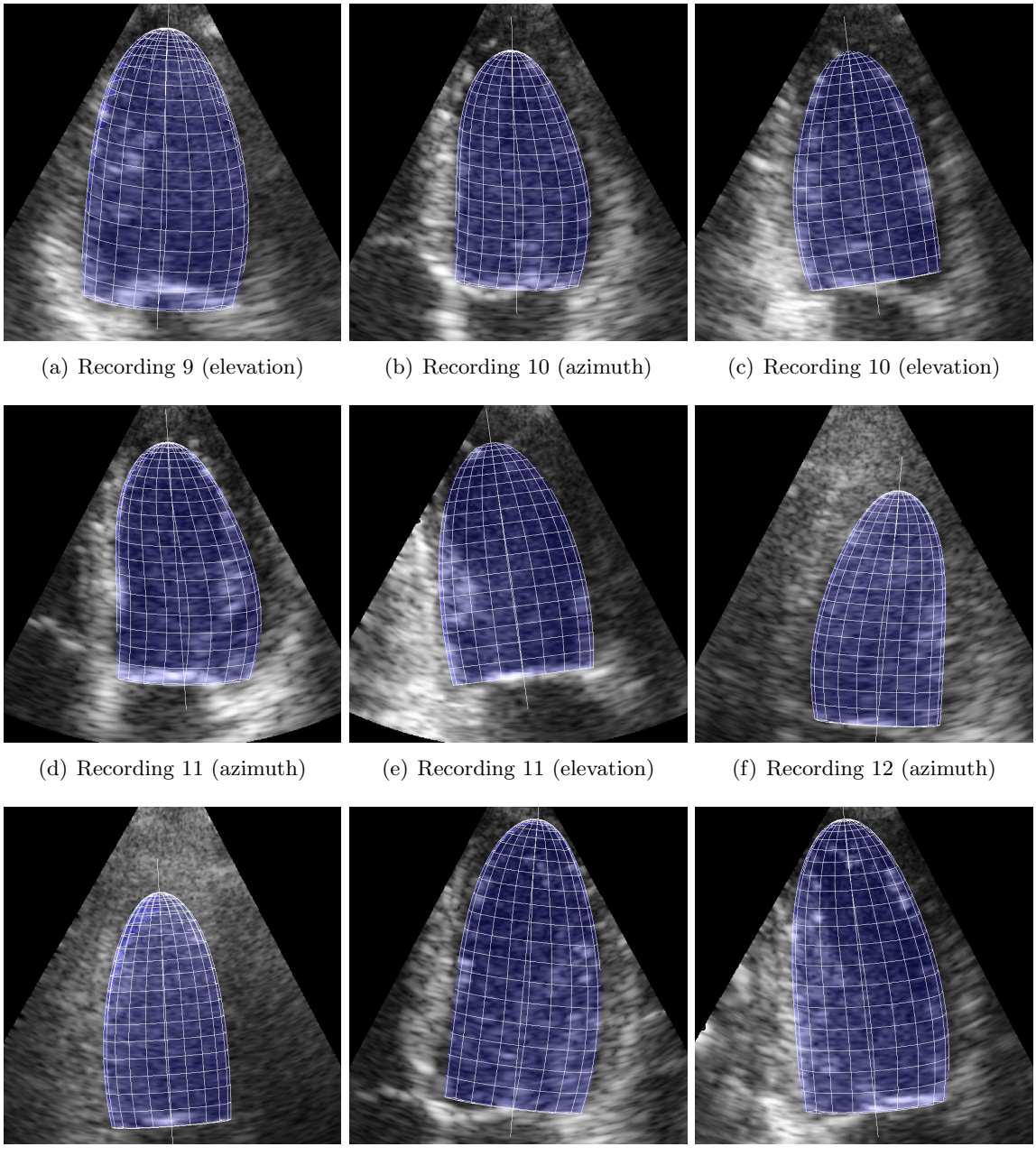


(a) Recording 5 (azimuth) (b) Recording 5 (elevation) (c) Recording 6 (azimuth)

(d) Recording 6 (elevation) (e) Recording 7 (azimuth) (f) Recording 7 (elevation)

(g) Recording 8 (azimuth) (h) Recording 8 (elevation) (i) Recording 9 (azimuth)

Figure 12.2: Screenshots from automatic tracking within the independent validation dataset (2/5).

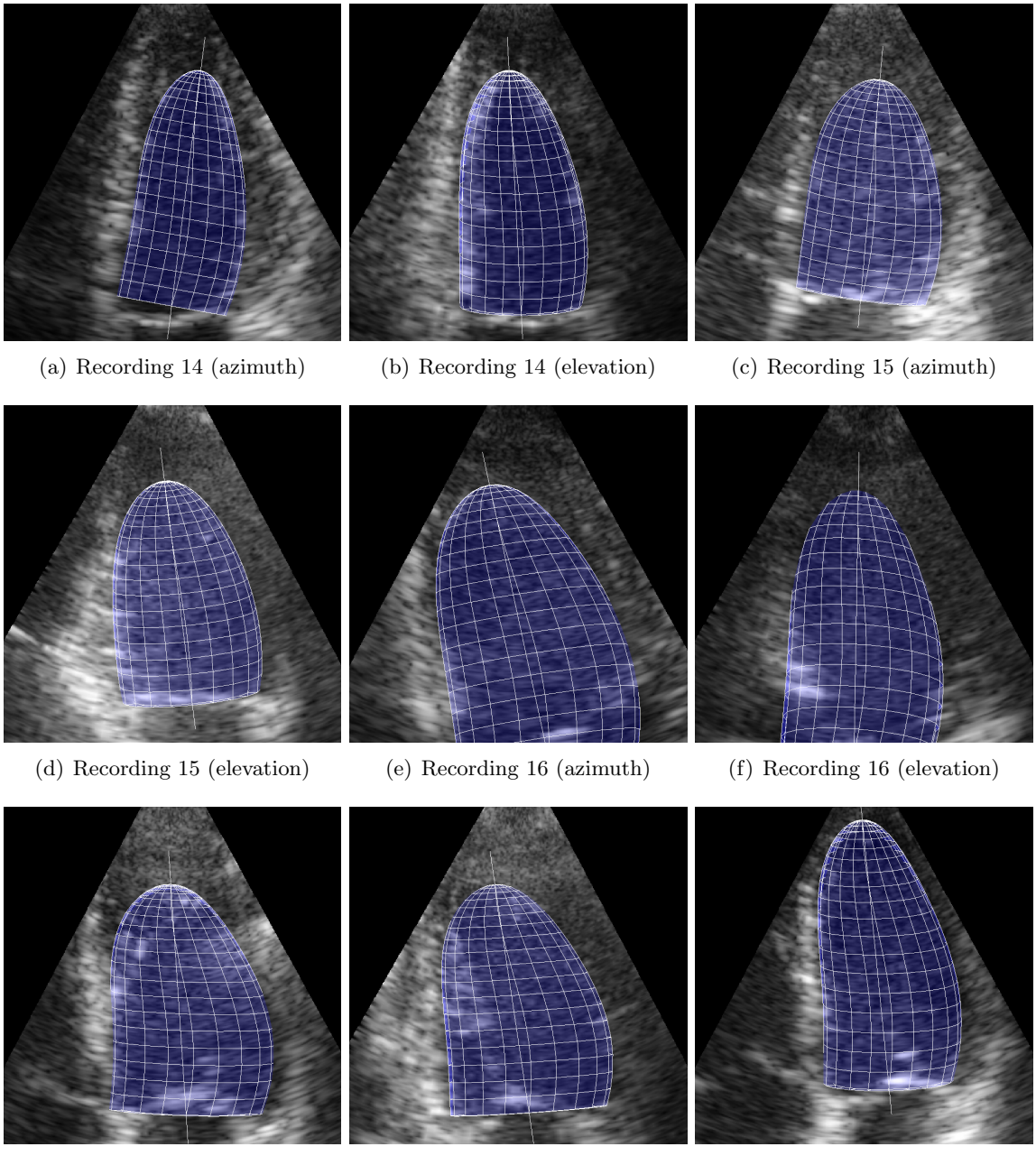


(a) Recording 9 (elevation) (b) Recording 10 (azimuth) (c) Recording 10 (elevation)

(d) Recording 11 (azimuth) (e) Recording 11 (elevation) (f) Recording 12 (azimuth)

(g) Recording 12 (elevation) (h) Recording 13 (azimuth) (i) Recording 13 (elevation)

Figure 12.3: Screenshots from automatic tracking within the independent validation dataset (3/5).

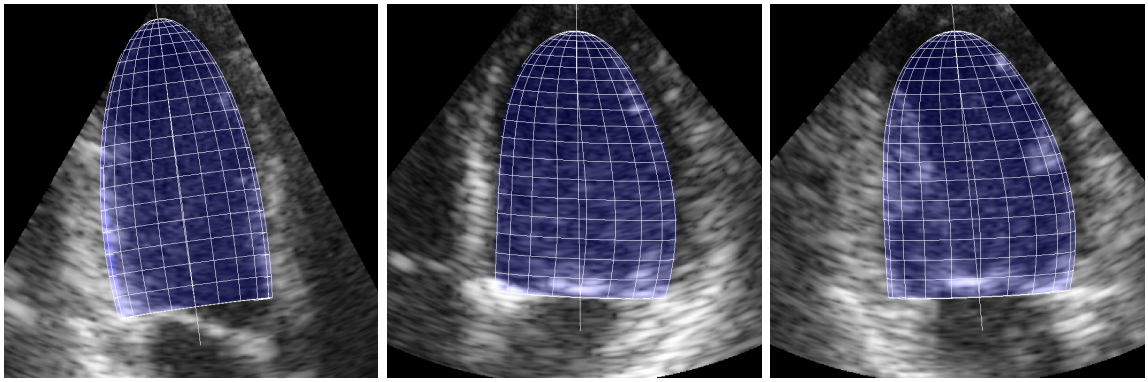


(a) Recording 14 (azimuth) (b) Recording 14 (elevation) (c) Recording 15 (azimuth)

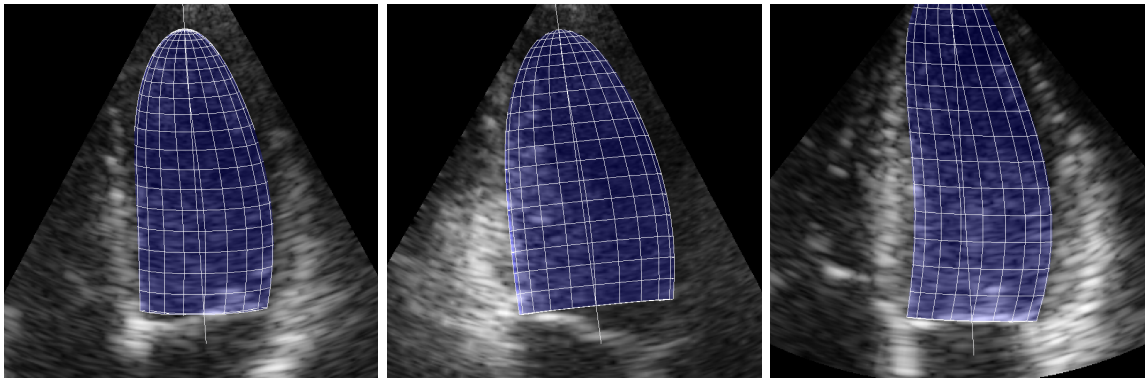
(d) Recording 15 (elevation) (e) Recording 16 (azimuth) (f) Recording 16 (elevation)

(g) Recording 17 (azimuth) (h) Recording 17 (elevation) (i) Recording 18 (azimuth)

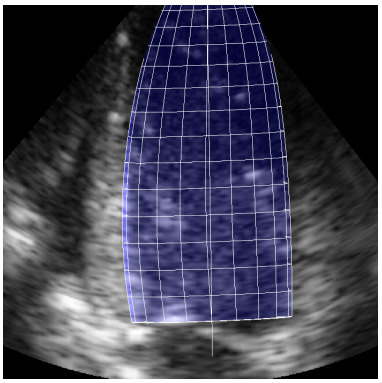
Figure 12.4: Screenshots from automatic tracking within the independent validation dataset (4/5).



(a) Recording 18 (elevation) (b) Recording 19 (azimuth) (c) Recording 19 (elevation)



(d) Recording 20 (azimuth) (e) Recording 20 (elevation) (f) Recording 21 (azimuth)



(g) Recording 21 (elevation)

Figure 12.5: Screenshots from automatic tracking within the independent validation dataset (5/5).

Chapter 13

Sensitivity to Initial Conditions Experiment

The goal of this experiment is to investigate how robust the tracking framework is to changes in the initial-contour used to initialize the tracking. Experiments are performed by translating the initial contour sideways along the three main axes, and matching the resulting contours with a non-translated reference contour .

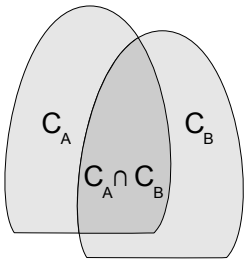


Figure 13.1: Illustration of two contours, $\mathbf{C}_A$ and $\mathbf{C}_B$, as well as their intersecting volume $\mathbf{C}_A \cap \mathbf{C}_B$. The union of the volumes $\mathbf{C}_A \cup \mathbf{C}_B$ are the total space covered by the contours, which equals $\mathbf{C}_A + \mathbf{C}_B - \mathbf{C}_A \cap \mathbf{C}_B$.

13.1 Matching Criteria

The matching is performed by calculating the ratio between the intersection and the union of the contour investigated $\mathbf{C}_I$, and the non-translated reference contour $\mathbf{C}_R$.

$$match \equiv \frac{\mathbf{C}_I \cap \mathbf{C}_R}{\mathbf{C}_I \cup \mathbf{C}_R}$$

This matching value will then always be limited to the range [0%, 100%]. An overall matching value is calculated by averaging the matching value for the tracked contours in all frames throughout the heart cycle.

13.2 Translation of Initial Contour Experiment

This experiment is performed by gradually translating the initial contour along all of the three main axes, and comparing the resulting tracked contours to a reference contour using the matching criteria described above. The reference contour used is simply the resulting tracked contour when the initial contour is not translated along any of the axes. The tracking is run for a couple of heartbeats, to let the contour settle down, before comparing the results.

Recording 10 from the independent validation dataset is used as basis for the tracking, which is a recording of high quality that has already been demonstrated to yield tracking of subjectively good quality.

Results

Table 13.1 shows the individual matching results for all experiments performed. The results are plotted in figure 13.2, which shows how gracefully the tracking accuracy degrades as the translation distance increases. Notice that tracking without translation also yields a match below 100%. This is due to the stochastic nature of the algorithm, where different runs of the same tracking setting can yield slightly different results.

Figure 13.2 clearly shows that the tracking for this particular recording is quite insensitive to the initial conditions. The initial contour can be translated almost 40mm in any direction without affecting the ability to track the left ventricle in any substantial way.

It is, however, strong reason to believe that echocardiography recordings of lesser quality will not yield the same robust behavior with regards to initial conditions, as the recording investigated here. This is because strong and consistent edge measurements are required to successfully guide the contour towards the correct location.

Offset	X-axis	Y-axis	Z-axis
-60mm	0%	0%	38%
-55mm	0%	0%	66%
-50mm	0%	0%	70%
-45mm	0%	0%	74%
-40mm	0%	0%	75%
-35mm	0%	71%	77%
-30mm	0%	75%	80%
-25mm	26%	74%	84%
-20mm	81%	84%	84%
-15mm	85%	88%	89%
-10mm	89&	91%	92%
-5mm	92%	94%	94%
0mm	93%	95%	95%
5mm	93%	94%	92%
10mm	87%	92%	89%
15mm	86%	82%	85%
20mm	81%	80%	83%
25mm	79%	78%	82%
30mm	75%	73%	79%
35mm	72%	68%	78%
40mm	0%	66%	74%
45mm	0%	0%	72%
50mm	0%	0%	70%
55mm	0%	0%	41%
60mm	0%	0%	23%

Table 13.1: Matching results from translating the initial contour in all three axes, and matching the resulting tracked contour with the tracked contour when the initial contour is not translated. The average matching percentage throughout the heart cycle is shown.

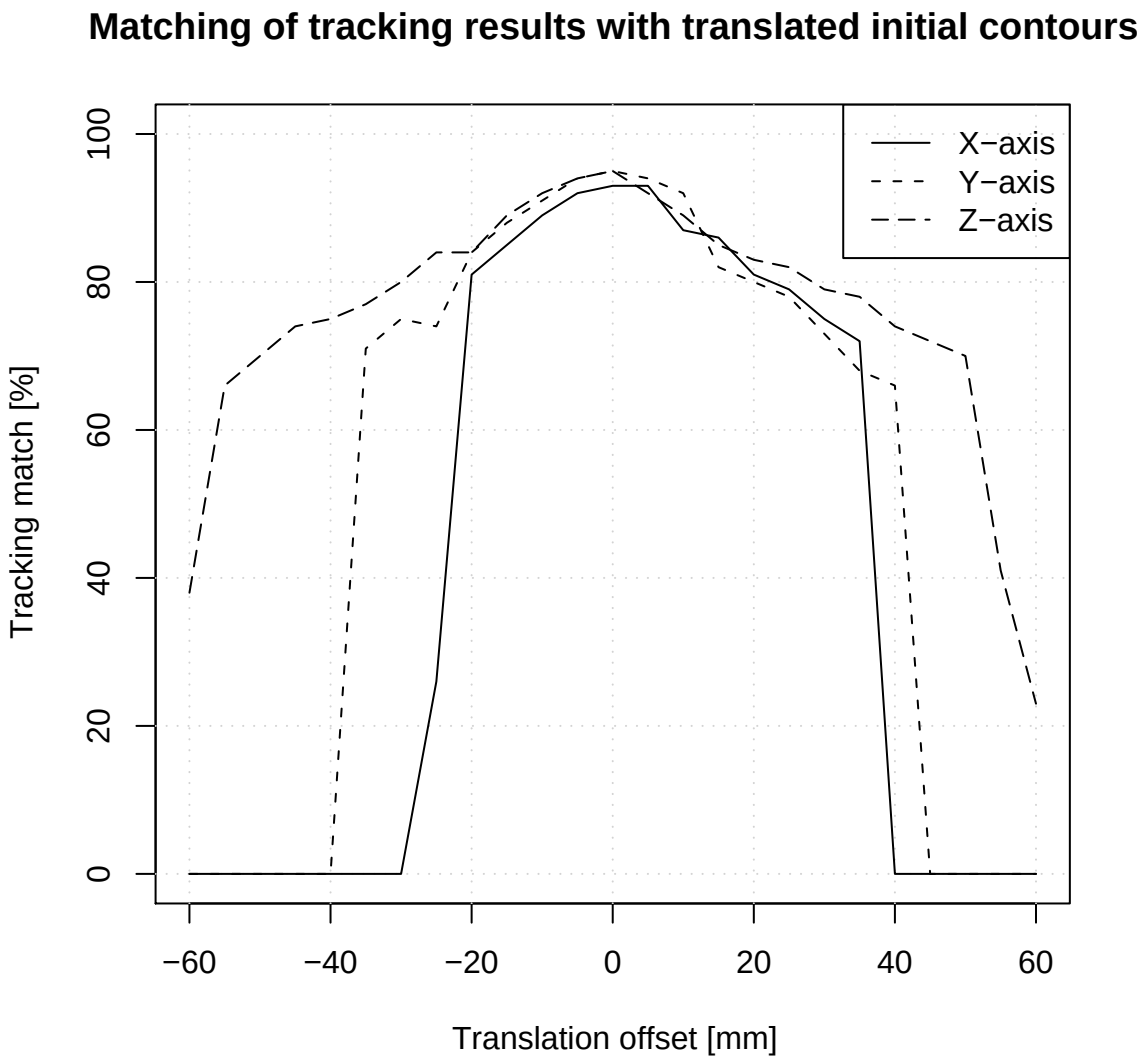
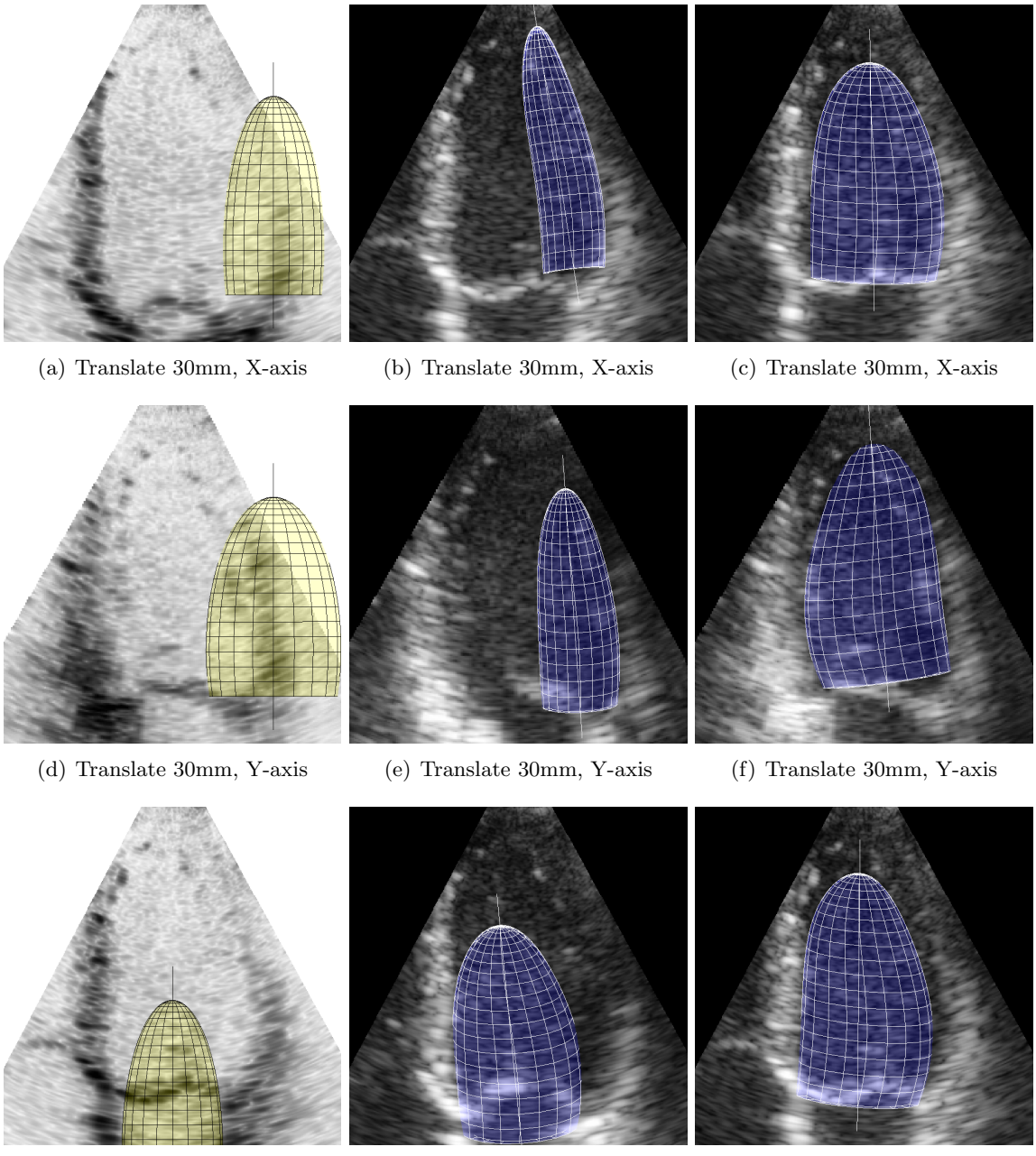


Figure 13.2: A plot of the tracking results shown in figure 13.1. The plot clearly shows how robust the tracking framework is to substantial translations of the initial contour.



(a) Translate 30mm, X-axis (b) Translate 30mm, X-axis (c) Translate 30mm, X-axis

(d) Translate 30mm, Y-axis (e) Translate 30mm, Y-axis (f) Translate 30mm, Y-axis

(g) Translate 30mm, Z-axis (h) Translate 30mm, Z-axis (i) Translate 30mm, Z-axis

Figure 13.3: Screenshots showing tracking with translated initial contours. The left column shows the translated initial contours, the center column shows the tracking after approximately a half heartbeat, and the right column shows the tracking after several heartbeats.

Chapter 14

Inference of Clinical Parameters Experiment

The aim of the clinical parameter experiment is determine the accuracy of diagnostic parameters automatically inferred from the tracking framework. These parameters include *end diastolic volume* (EDV) *end systolic volume* (ESV) and *ejection fraction* (EF), which are all easily calculated based on the estimated state vector of the ventricle deformation model.

14.1 Inference of Clinical Parameters

The proposed LV-tracking framework opens up the possibility for automatic real-time ventricle volume and ejection fraction calculations. This is fundamentally simple, since all deformation parameters, except scaling, are volume preserving. The ventricle volume is thus proportional to the product of s_x, s_y and s_z. End diastolic volume and end systolic volume are found by determining the maximum and minimum contour volume throughout the heart cycle.

$$\begin{aligned} vol &\propto s_x\, s_y\, s_z \\ EDV &= \max\{vol\} \\ ESV &= \min\{vol\} \\ EF &= \frac{EDV - ESV}{EDV} \end{aligned}$$

14.2 Clinical Parameter Experiment

The evaluation is based on comparison between automatic tracking and manual measurements on 19 of the recordings in the GE Vingmed Ultrasound dataset where manual EDV/ESV/EF-measurements were present. The rest of the 88 recordings in the dataset lacked manual measurement, and were therefore excluded from the comparison. A special exclusion were also made for recording 22, where the automatic tracking failed so miserably that automatic failure detection would have been trivial.

WARNING: This is the same dataset as was used to develop and tune the tracking algorithm. Results may therefore be better than what can be expected in a random dataset.

Results

Figures 14.1 and 14.2 clearly shows that the automatic tracking suffers from severe bias. The tracking consistently returns overestimates for low volume measurements. The bias does, however, decrease as the contour volume increases.

Linear regression on the results in figure 14.1 and 14.2 shows that the approximation error is not that bad, with $R^2 = 0.8054$ for EDV and $R^2 = 0.9269$ for ESV. This means that the parameter bias probably can be corrected by simple subtraction followed by a normalizing multiplication.

The case for EF is unfortunately a bit worse. Figure 14.3 shows a significantly larger approximation error, with $R^2 = 0.6590$ for EF. It is therefore clear that the biases in EDV and ESV do not cancel each other out. Instead they seem to build on each other.

	Manual segmentation			RCTL tracking		
Recording no	EDV	ESV	EF	EDV	ESV	EF
Recording 5	108	64	41	116	80	31
Recording 6	84	44	48	118	72	39
Recording 7	124	46	63	134	62	54
Recording 8	112	83	26	109	86	21
Recording 12	155	56	64	125	70	44
Recording 16	82	29	65	105	55	48
Recording 23	72	40	44	94	59	37
Recording 27	56	13	77	81	56	31
Recording 28	127	101	21	149	115	23
Recording 30	74	59	20	106	92	13
Recording 31	141	73	48	139	79	43
Recording 32	168	58	65	132	85	36
Recording 33	135	103	24	134	103	23
Recording 37	109	62	43	116	79	32
Recording 38	110	62	43	117	79	32
Recording 39	234	202	14	188	160	15
Recording 41	187	159	15	146	131	10
Recording 46	170	114	33	142	103	27
Recording 52	55	22	60	55	37	33

Table 14.1: Results of clinical evaluation, showing *end diastolic volume* (EDV), *end systolic volume* (ESV) and *ejection fraction* (EF) for manual tri-plane segmentation and automatic tracking using the real-time contour tracking library (RCTL).

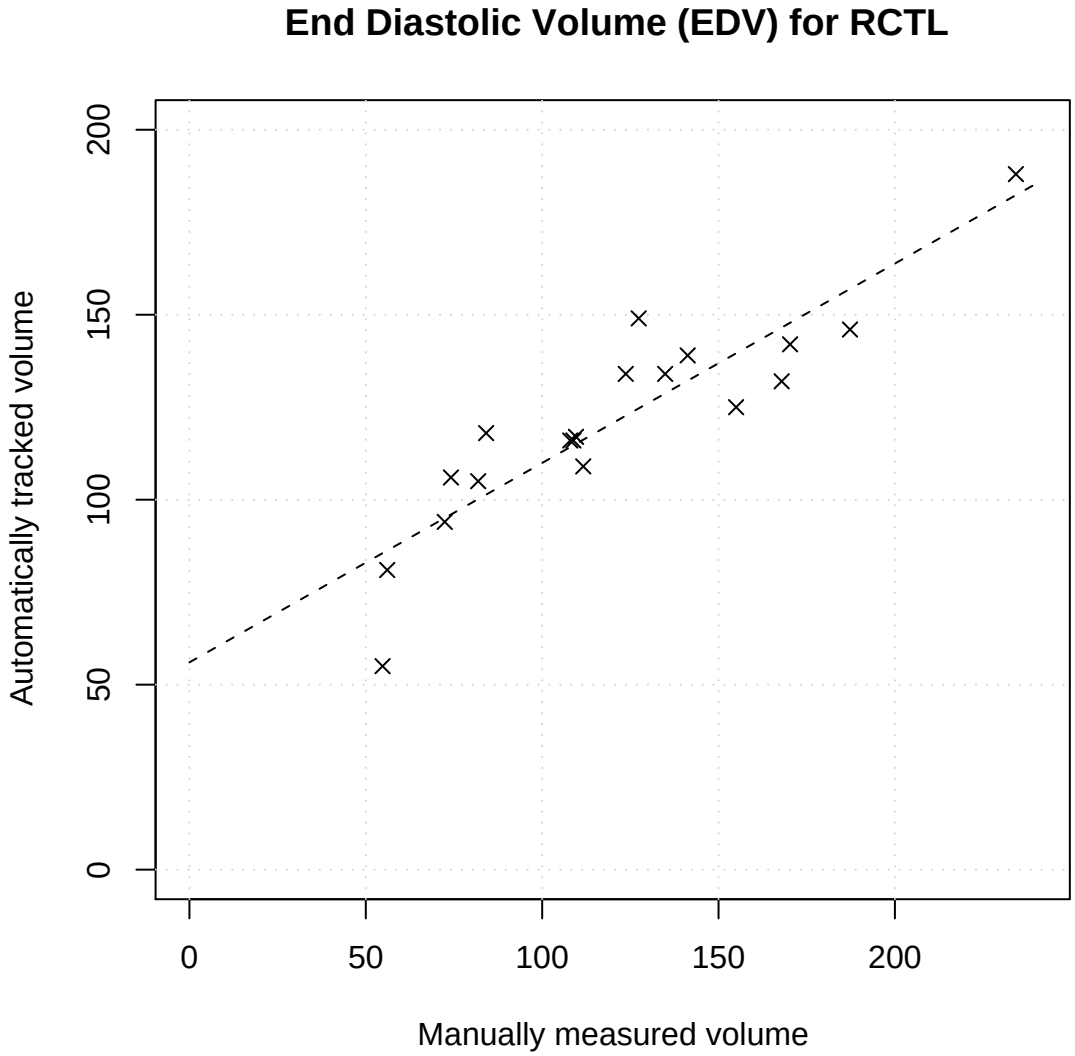


Figure 14.1: End diastolic volume (EDV) relationship between real-time contour tracking and manual tri-plane segmentation. Linear regression yields the line $y = 0.539\,x + 56.029$ with $R^2 = 0.8054$.

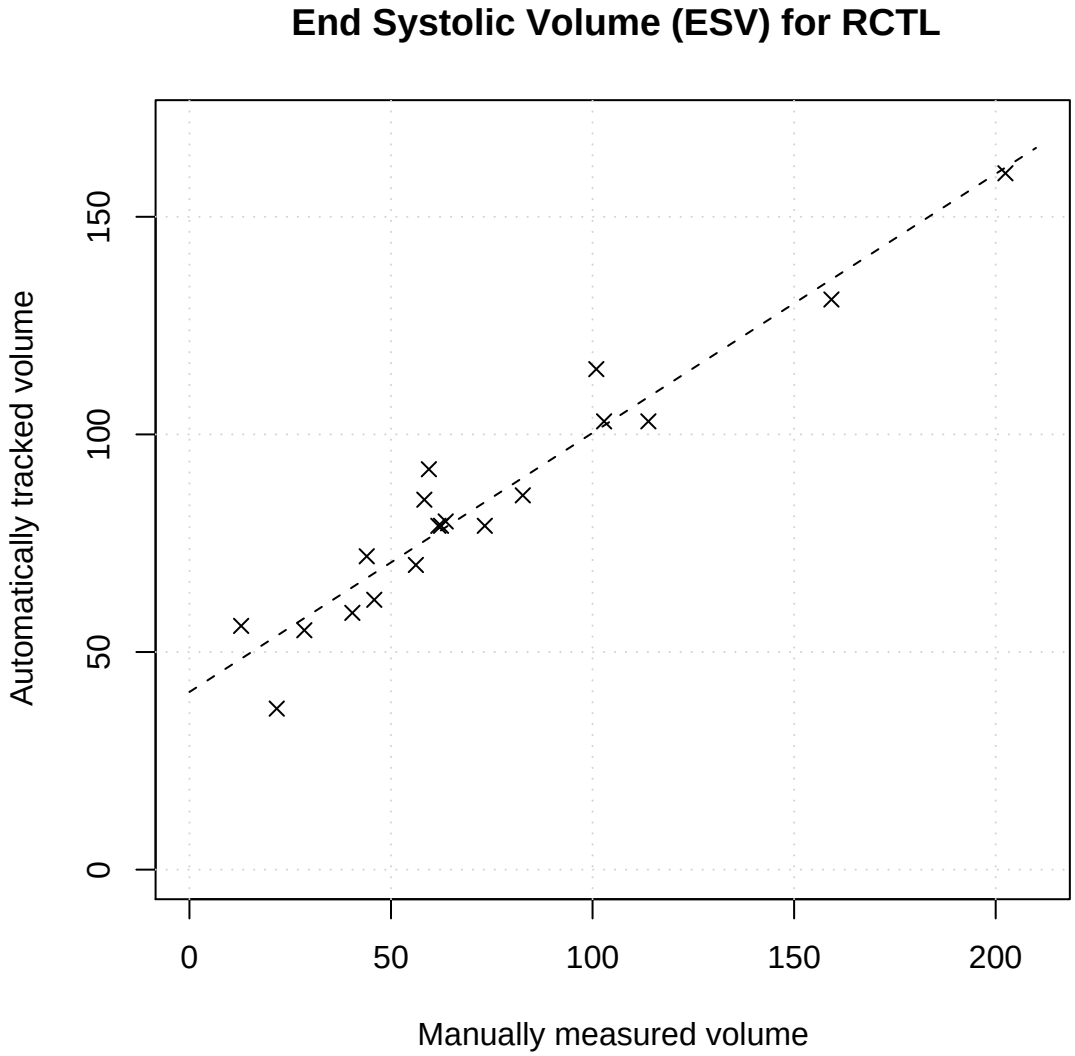


Figure 14.2: End systolic volume (ESV) relationship between real-time contour tracking and manual tri-plane segmentation. Linear regression yields the line $y = 0.5953\,x + 40.815$ with $R^2 = 0.9269$.

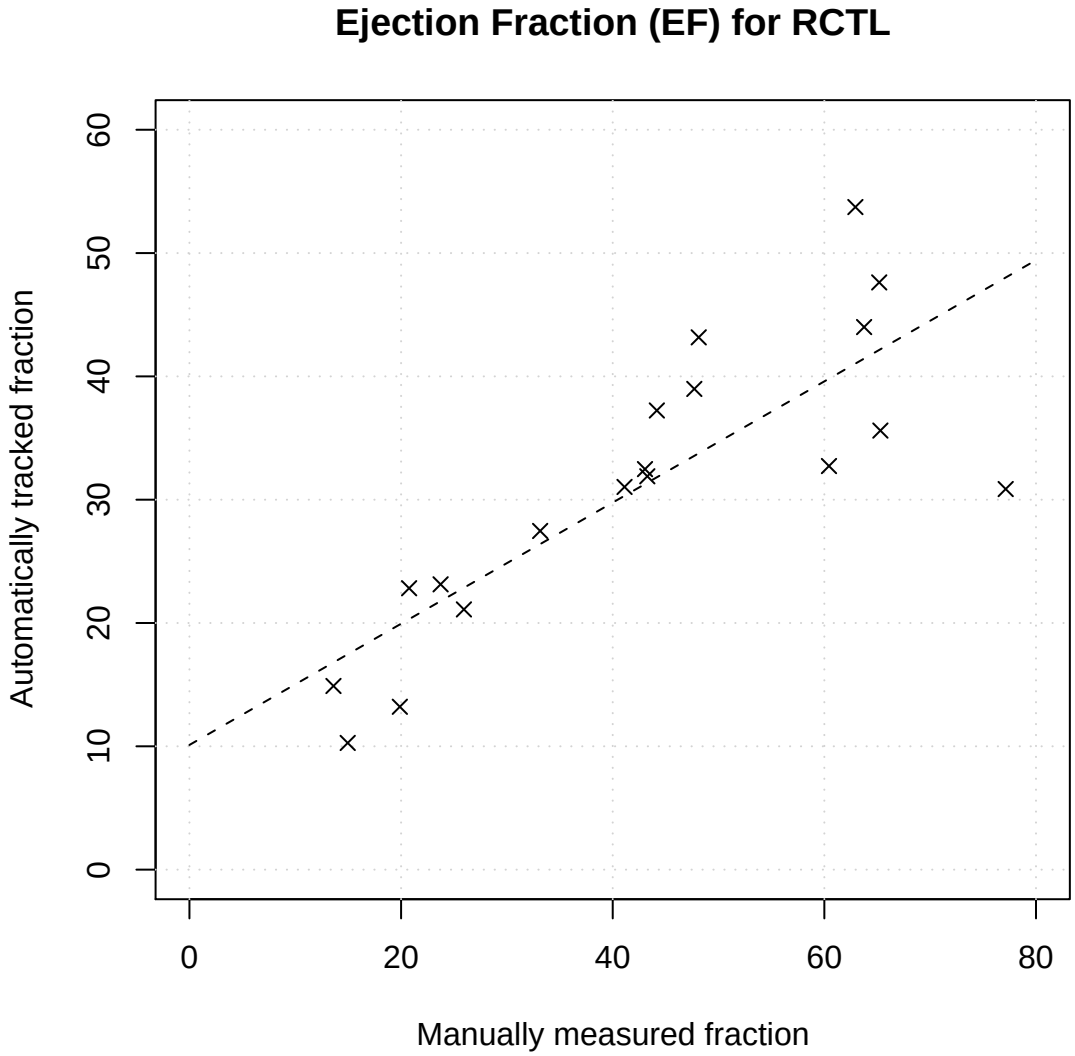


Figure 14.3: Ejection fraction (EF) relationship between real-time contour tracking and manual tri-plane segmentation. Linear regression yields the line $y = 0.4914\,x + 10.112$ with $R^2 = 0.6590$.

Chapter 15

Running Time Analysis Experiment

One of the main advantages of the tracking framework proposed is its ability to perform tracking in real-time. The experiments performed using a 10-parameter LV-model with 426 contour points and search normals consisting of 20 samples in 25fps 3D echocardiography datasets only yielded a modest CPU load of approximately 18% for real-time tracking[1].

Scanconversion and edge detection greatly dominate the running time. Simultaneous visualization of several scanplanes together with the tracking greatly increases the processing requirements due to excessive scanconversions, leading to a maximum framerate of approximately 10fps.

Figure 15.1 shows that computational complexity scales linearly with the number of contour points used. One clearly sees that edge detection strongly dominate the running time, since the processing time for zero contour points is almost zero. The running time of the tracking can therefore easily be tuned by adjusting the number of contour points used.

Figure 15.2 shows that computational complexity also scales roughly linearly with the number of deformation parameters within the interval investigated. It is reason to believe that the curve eventually becomes quadratic or cubic, as the number of parameters increases well above the interval investigated. The estimate covariance matrices used in the Kalman filter and summed measurement covariance matrices used in edge detection have dimensions equal to the number of parameters, leading to a quadratic number of matrix elements. Increased running time then emerges, because matrix operations like addition, multiplication and inversion generally have quadratic or cubic time complexity. The number of deformation parameters required for this nonlinear effect to become significant is, however, unknown, and will be in focus for further research in this area.

[1]The tracking were then performed on a 3GHz Intel Pentium 4 processor, with visualization disabled.

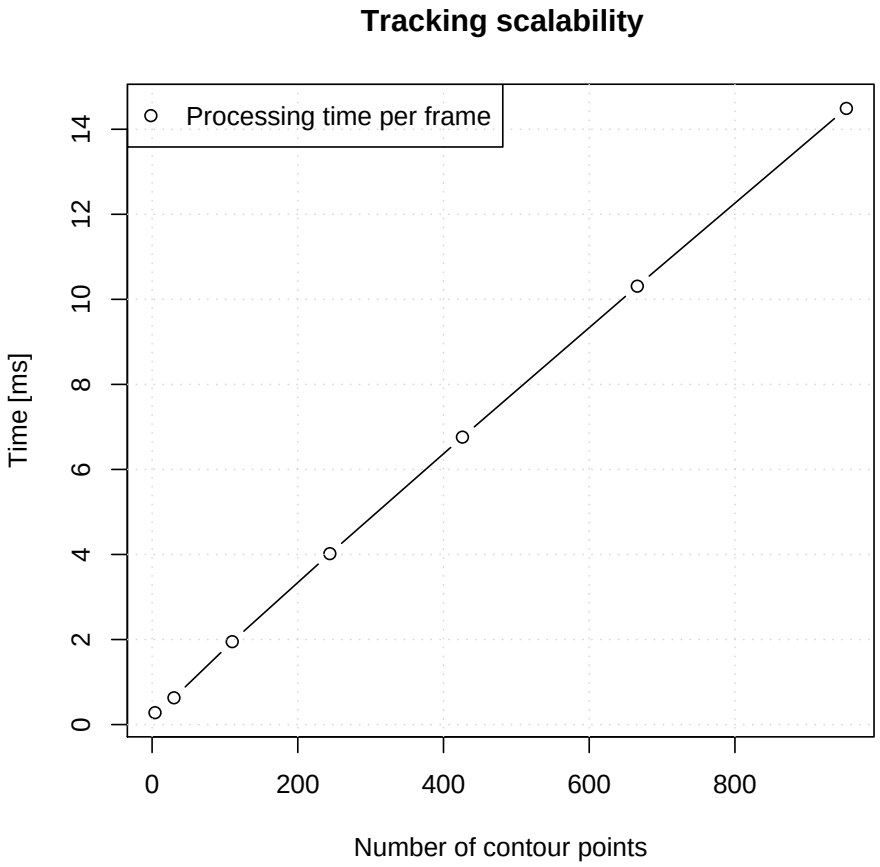


Figure 15.1: Tracking scalability for the LV-deformation model with 10 parameters.

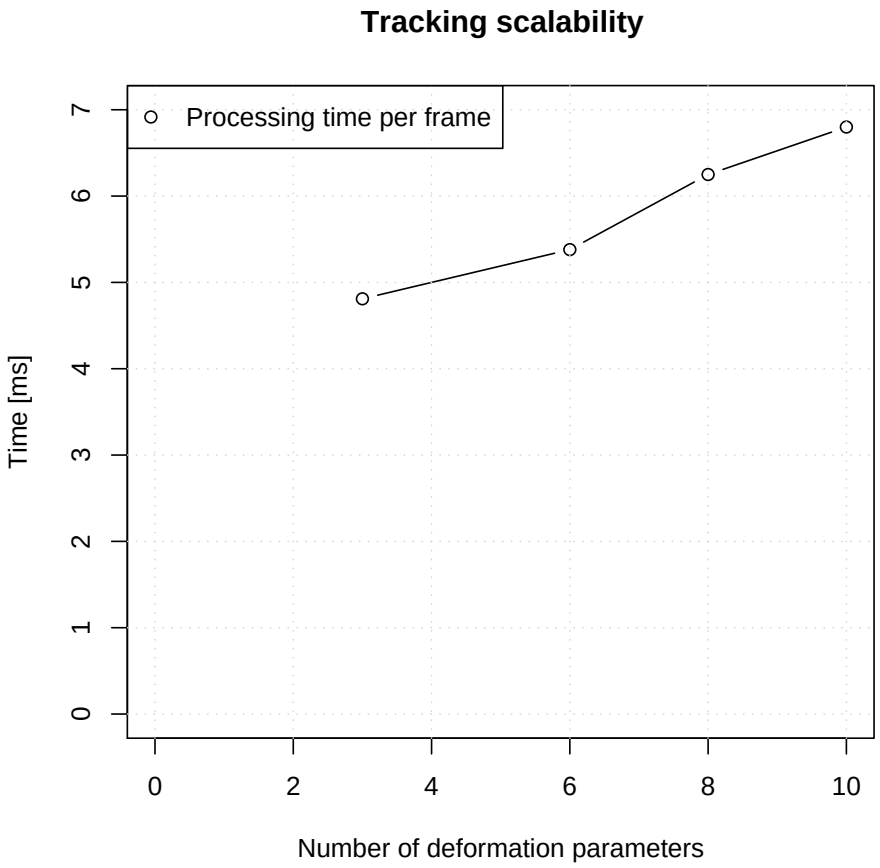


Figure 15.2: Tracking scalability for contours consisting of 426 points. Benchmarking is performed by gradually simplifying the 10-parameter LV-model proposed in the thesis. Bending (c_x, c_y) is first removed, then rotation (r_x, r_y), and finally also scaling (s_x, s_y, s_z), leading to a deformation model consisting of only translations (t_x, t_y, t_z).

Part IV

Discussion & Conclusion

Chapter 16

Discussion

Usage of the tracking method proposed in this thesis on several datasets yielded promising results for automatic real-time left ventricular tracking in 3D echocardiography. Comparison between the results from the training dataset and the independent validation datasets indicate that the method shows good generalization properties, and does not rely on laborious parameter tuning to function well on each of the recordings.

It can be argued that the evaluation procedure performed is too subjective and should have been performed by a medical clinician. This is of course true, but the objective was not to prove that the method is very accurate or mature enough for clinical usage. The principal objective was merely to evaluate the *feasibility* of the method, which I think has been clearly demonstrated. Further research will focus on perfecting the method, and striving towards tracking that is both robust and accurate.

Experimental results suggest that the tracking is fairly robust to changes in the initial parameters when the image quality is good. The degree of robustness for recordings of lesser quality is however unknown, but believed to be significantly poorer, since strong and consistent edge measurements are required to successfully guide the contour towards the correct location.

Investigation of automatic inference of clinical parameters directly from the tracking yielded results that were severely biased. Much further effort must therefore be laid down to increase tracking accuracy before one can expect diagnostic parameters of clinical value. Usage of alternative algorithms for sequential state estimation, like the iterated Kalman filter and particle filters might very well lead to improved results here.

The tracking framework is also the only known implementation of real-time contour tracking using deformable models in a 3D dataset[1]. Most prior art have used optimization algorithms for tracking that requires many iterations to converge, leading to running times significantly higher than what is required for real-time processing. The running-time is linear in the number of contour points used, which makes it easy to scale the tracking performance.

The discussion continues in the *Research Plan* appendix, which focuses on implications and

[1]Neither I, nor anyone in the GE Vingmed Ultrasound development team has been able to discover any prior publications on real-time contour tracking in 3D datasets.

plans for future research, based on the results found in this thesis.

Chapter 17

Conclusion

A novel framework for real-time contour tracking in 3D video using sequential state estimation has been proposed. The framework builds upon previous work by Blake et al., and places very few constraints on the type of contour and deformation models allowed.

The feasibility of using this tracking framework for real-time left ventricular tracking in 3D ultrasound has been demonstrated by automatic testing in several datasets. The tracking framework was found to automatically detect LV position initially, even in situations where it is partially outside the volume or the initial contour were inaccurately placed. It also successfully tracks the dominant motion throughout the heart cycle, and correctly identifies the long-axis.

Primary scientific contributions:

- Extension of existing contour tracking framework (Blake et al.) to tracking in 3D-data.
- Generalization of existing contour tracking framework (Blake et al.) to all types of contours, deformed by arbitrary, nonlinear deformations.
- Derivation of, and extensions to the information filter/sequential estimation equations by Blake et al. to allow more efficient processing of independent measurements.
- Experimental validation of feasibility of real-time left ventricular tracking in 3D echocardiography.

Part V

Appendices

Appendix A

Real-time Contour Tracking Library

Real-time Contour Tracking Library (RCTL) is a video tracking library, written in C++, and developed in conjunction to this thesis. It is capable of tracking a wide range of contour models deformed by nonlinear deformation models in both 2D and 3D video. Most focus have been on head-tracking using a web-camera, and LV-tracking in 3D-echocardiography, however support for other object types and imaging modalities can easily be added.

GE Vingmed Ultrasound AS has purchased a license for exclusive commercial usage of the library, and the library will be integrated with the *Vivid*-series of GE Healthcare cardiovascular ultrasound scanners for online testing within the next few months.

The agreement for commercialization of the library in collaboration with GE Vingmed Ultrasound unfortunately prevents me from attaching any source code to this thesis. Only simple figures showing the modular decomposition of the library can therefore be presented.

A.1 Modular Decomposition

The library is written in a modular fashion, with clearly defined interfaces for video sources, contour models, deformation models and edge models. The tracking algorithms are also somewhat modular in their structure. Extensions to the library can thus be performed quite easily.

All linear algebra is performed using the *Boost uBlas* library, which is supplemented with home brewed code for matrix factorization, matrix inversion, as well as sampling and likelihood calculations for multivariate normal distributions. The *Simple Directmedia Layer* (SDL) library is used for windows handling and the *OpenGL* library for visualization and 3D-rendering.

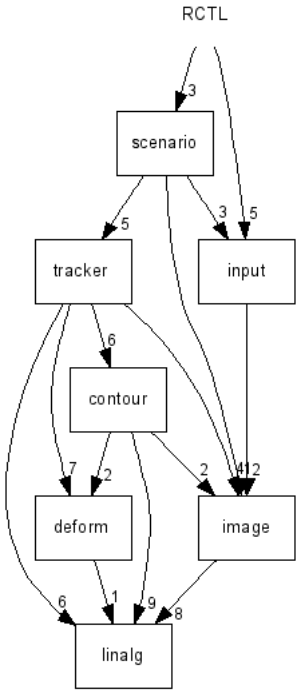


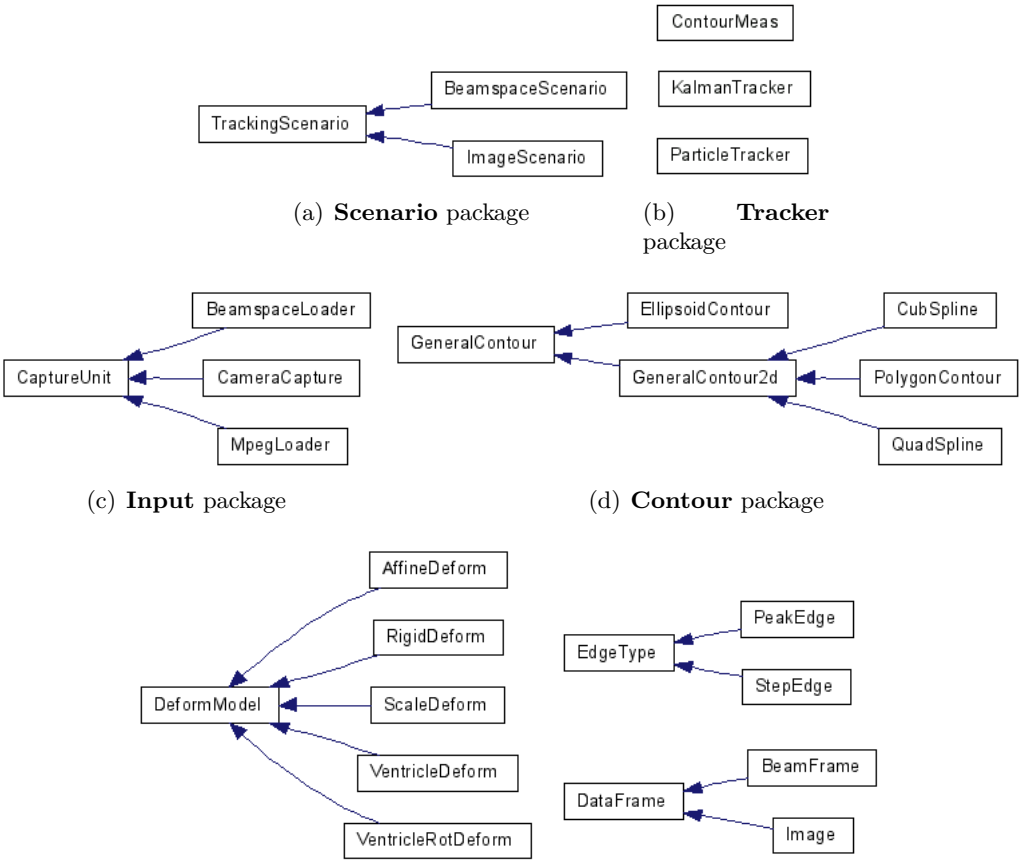
Figure A.1: Module dependency graph for *Real-time Contour Tracking Library* (RCTL).

A.2 Configuration Files

The library uses configuration files written in XML to specify:

- Input type and source
- Edge detection model and parameters
- Contour model and initial/prior state
- Tracking algorithm and parameters
- Kinematic model parameters for regularization and inertial/damped motion

The configuration files are typically used to tune tracking parameters, like initial contour location, edge-detection sensitivity, and kinematic parameters.



(a) **Scenario** package (b) **Tracker** package

(c) **Input** package (d) **Contour** package

(e) **Deform** package (f) **Image** package

Figure A.2: Primary content of each package within RCTL. Classes are shown using inheritance graphs.

Appendix B

Research Plan

As a student attending the *integrated PhD program* I am almost halfway through my time as a PhD student at the faculty of Information technology, Mathematics and Electronics (IME), at the Norwegian University for Science and Technology (NTNU). This thesis does therefore also serve as a research plan for the remaining part of my PhD. I have chosen to categorize my plans into two sections: *technical plans*, for fundamental technical and theoretical challenges associated with real-time contour tracking, and *clinical applications plans*, for potential areas of application for the tracking technology. Both areas are of clear scientific interest, although the potential application areas undoubtedly is the section of greatest industrial interest, especially within the field of medical imaging. The chapter concludes with a presentation of my publication plans.

B.1 Technical Focus Areas

The following technical focus areas have been identified as especially important for my future research:

- Improvement to the deformation models, by incorporating models developed within the field of *biomechanics*.
- Extensions for tracking of local deformations in addition to global tracking. Typically by means of PCA-models.
- Improvement to the edge detection, by letting the edge models vary across the contour.
- Methods to exploit the correlation between nearby measurement vectors for improved robustness.
- Monte Carlo extensions to the tracking algorithm to improve robustness at the expense of running-time.
- Tracking of contour "twisting" using point-tracking.

Combine With Local Tracking

The tracking framework does currently mainly perform tracking using a quite rigid and global LV contour. Extensions to also track local deformations, possibly by focusing on deviations

from the global contour, can probably be interesting. This might be useful in detecting heart failures.

Point tracking

A fundamental limitation associated with the *normal displacement* approach to contour tracking in this thesis is its inability to track contour "twisting". Twisting is movement planar to the contour surface, and does thus not induce any normal displacements.

Tracking of twisting is often not considered especially important, but may none the less be important for some applications, like echocardiography, where twisting can be used to analyze muscular contractions within the heart. This is already done by tracking selected points on the cardiac wall throughout the heart cycle in 2D ultrasound, using a point tracking technique known as *speckle tracking*, where movement in the speckle pattern found in all ultrasound recordings is being tracked. The tracking results are then presented to the user as *strain rate* images, where muscular contractions are color-coded based on stiffness and elasticity.

Extensions of the tracking framework for point tracking in 3D would therefore be of great interest. Some research has already been done in this field for using point tracking in connection to probabilistic contour tracking. A recent approach of particular interest is Comaniciu et al. in [11] and [36], who focused on real-time point tracking in long-axis 2D echocardiography, using a Kalman-filter framework which exploited the heteroscedastic property of measurement uncertainty. This means that the edge detection accuracy is not equal in all directions. These papers does, however, track individual contour points, and does not use a mathematical deformation model.

It would be interesting to pursuit the approach by Comaniciu et al., and try to connect point tracking to the existing tracking framework. This would, however, probably lead to increased computational complexity, since the search space for point tracking in 3D is much larger compared to the 1D search space for normal displacements.

Use as Input for More Accurate Volume-segmentation

The tracking results can also be used as input for more accurate volume-segmentation algorithms. These algorithms are typically formulated as constrained optimization problems, requiring iterative evaluation for gradual convergence.

B.2 Clinical Focus Areas

Future research in collaboration with GE Vingmed Ultrasound will focus on exploring the tracking framework's potential for clinical applications. A number of possibilities have already been identified, and are described below.

Operator Guidance

The tracking framework can probably be used to give a visual indication on where the left ventricle is located within the volume, and warn the operator if the ventricle disappears partly

outside the volume. This can also be combined with a warning if the apex disappears outside of the volume due to unconscious sliding of the probe on the patients' chest.

Ventricle Volume and Ejection-Fraction

Preliminary experiments in this thesis have explored the possibility of automatic diagnostic parameters inferred from the tracking. Accurate volume and ejection-fraction measurement will, however, require tracking results of high accuracy in order to be useful. It is still unknown at this state whether the tracking can be made accurate enough to yield volume and EF calculations of clinical significance, but much effort will be laid down to improve tracking accuracy using alternative algorithms and adjusted models.

Automatic LV Alignment

Successful LV-tracking automatically yield the position and orientation of the left ventricle. These parameters can be used to automatically translate and rotate the dataset to center and stabilize the left ventricle image. This would probably make it easier for clinicians to study regional deformations within the heart.

Automatic 9-Slice Alignment

GE Healthcare already has an imaging modality called *9-slice*, where nine equidistant short-axis views of the left ventricle are displayed simultaneously. This modality does, however, require the user to manually select ventricle position and orientation prior to examination or user presentation. Automatic usage of ventricle position and orientation from the LV-tracking can probably be used to make 9-slice fully automatic. Usage of the scaling parameters from tracking can also be used to automatically adjust the distance between the 9-slice planes, thus avoiding any out-of-plane movement.

Simultaneous Endo- and Epicardial Tracking

Simultaneous endo- and epicardial tracking can be performed by using a thick-wall model for the ventricle. This may yield both more robust tracking, as well as open up the possibility for calculation of wall-thickening throughout the cycle. Automatic calculation of left ventricular mass may also be possible using this approach.

B.3 Industry Collaboration

This thesis have been written in collaboration with GE Vingmed Ultrasound, who provided me with clinical applications for the tracking technology presented within the field of 3D echocardiography, as well as valuable feedback throughout the thesis work. This company has also purchased a commercial license to the software library developed, as well as signed a contract for partly financing the remaining part of my PhD. Further research will therefore continue to be done in close collaboration with Vingmed Ultrasound to ensure both excellence in science and high industry relevance.

Stein Inge Rabben at GEVU will also take over responsibility for supervising my research for the remaining part of my PhD, in addition to my current supervisor Bjørn Olstad.

B.4 Publication Plans

The author have already published the paper:

- *Comparison of Kalman Filter Estimation Approaches for State Space Models with Nonlinear Measurements*, Scandinavian Conference on Simulation and Modeling - SIMS 2005

which compares algorithms for sequential state estimation in models with nonlinear measurements, which is exactly the case for the contour tracking framework presented in this thesis. This paper does therefore deal with one of the fundamental problems encountered in real-time contour tracking.

Main publication goals for the tracking framework presented in this thesis:

- Feasibility article for real-time LV-tracking in *Computers in Cardiology [accepted]*
- Reference article in *IEEE Transactions on Medical Imaging [planned]*
- Clinical article on automatic Ejection-Fraction estimation in a clinical journal *[planned]*

Abstract for the feasibility article have already been accepted for publication in *Computers in Cardiology 2006*, which will be partly based on the theory and results developed in this thesis. Work on the reference article will begin this fall, and will probably go on for some time while the theories mature and results improve. Regarding a clinical article, I have already begun contacting potential clinicians in cardiology for initiating a medical study on automatic ejection fraction estimation. There is, however, a bit more unclear how long time it will take before such a study can begin.

In addition to the main publication goals listed above, I will also strive towards research results worthy of publication in both the technical and clinical focus areas listed above. I particularly expect research regarding tracking robustness and improved deformation models to be fruitful and result in publishable results, as well as the attempted approaches towards automatic diagnosis.

Appendix C

Results from Automatic Tracking in the Training Dataset

Detailed tracking results		
Recording no	Tracking result	Comments
Recording 1	n/a	File missing.
Recording 2	n/a	File missing.
Recording 3	n/a	File missing.
Recording 4	good	Good tracking in poor contrast data.
Recording 5	good	Excellent tracking.
Recording 6	adequate	Some problems tracking apex.
Recording 7	good	Excellent tracking.
Recording 8	good	Robust tracking, even in presence of model mismatch.
Recording 9	adequate	Problems tracking apex and one wall due to poor contrast.
Recording 10	fair	Very poor contrast and sensitive for input contour.
Recording 11	good	Good tracking, even in presence of model mismatch.
Recording 12	adequate	Problems tracking the long-axis.
Recording 13	good	Good tracking in contrast recording.
Recording 14	good	Good tracking in contrast recording.
Recording 15	good	4-chamber view
Recording 16	good	Excellent tracking.
Recording 17	fair	Parasternal view, volume largely missing.
Recording 18	fair	Parasternal view, volume largely missing.
Recording 19	fair	Parasternal view, poor wall contrast on one side.
Recording 20	poor	4-chamber view with LV almost completely outside the volume.
Recording 21	good	4-chamber view with one wall outside the volume.
Recording 22	poor	Very small LV. Initial contour too large and deep.
Recording 23	adequate	Small LV, but fairly good tracking.
Recording 24	good	Poor wall contrast on one side, but good tracking.
Recording 25	fair	Problems tracking apex.
Recording 26	n/a	Corrupt file.
Recording 27	good	Excellent tracking.
Recording 28	good	Apex outside volume, but good tracking.
Recording 29	n/a	Corrupt file.
Recording 30	adequate	Problems tracking apex.
Recording 31	good	Good tracking in low contrast.

Table C.1: Results of automatic tracking within the GE Vingmed Ultrasound dataset (1/3).

Detailed tracking results		
Recording no	Tracking result	Comments
Recording 32	fair	Problems tracking apex.
Recording 33	good	Good tracking.
Recording 34	poor	Contrast recording with almost no wall contrast.
Recording 35	fair	Contrast recording, problems tracking mitral plane.
Recording 36	fair	Contrast recording with poor contrast.
Recording 37	good	Good tracking, but does not reach apex due to model mismatch.
Recording 38	good	Good tracking.
Recording 39	good	Excellent tracking in tilted volume.
Recording 40	adequate	Poor wall contrast on one side.
Recording 41	good	Good tracking, but a bit sensitive for initial contour.
Recording 42	poor	Problems reaching mitral plane.
Recording 43	adequate	Strong echo artifacts near apex.
Recording 44	adequate	Contrast recording. Mitral plane problems.
Recording 45	adequate	Strong echo from cavity. One wall outside volume.
Recording 46	good	Some model mismatch, but good tracking.
Recording 47	adequate	Contrast recording with very low contrast.
Recording 48	good	Excellent tracking.
Recording 49	good	Good tracking in poor contrast.
Recording 50	good	Good tracking.
Recording 51	adequate	4-chamber view with poor image quality near apex.
Recording 52	good	Good tracking.
Recording 53	good	Poor wall contrast on one side, but good tracking.
Recording 54	good	Good tracking.
Recording 55	fair	Poor contrast.
Recording 56	good	Poor contrast.
Recording 57	good	Excellent tracking.
Recording 58	fair	Problems tracking apex.
Recording 59	good	Poor image quality.
Recording 60	good	Excellent tracking.
Recording 61	adequate	Parasternal view, volume partly missing.
Recording 62	good	Poor wall contrast on one side.

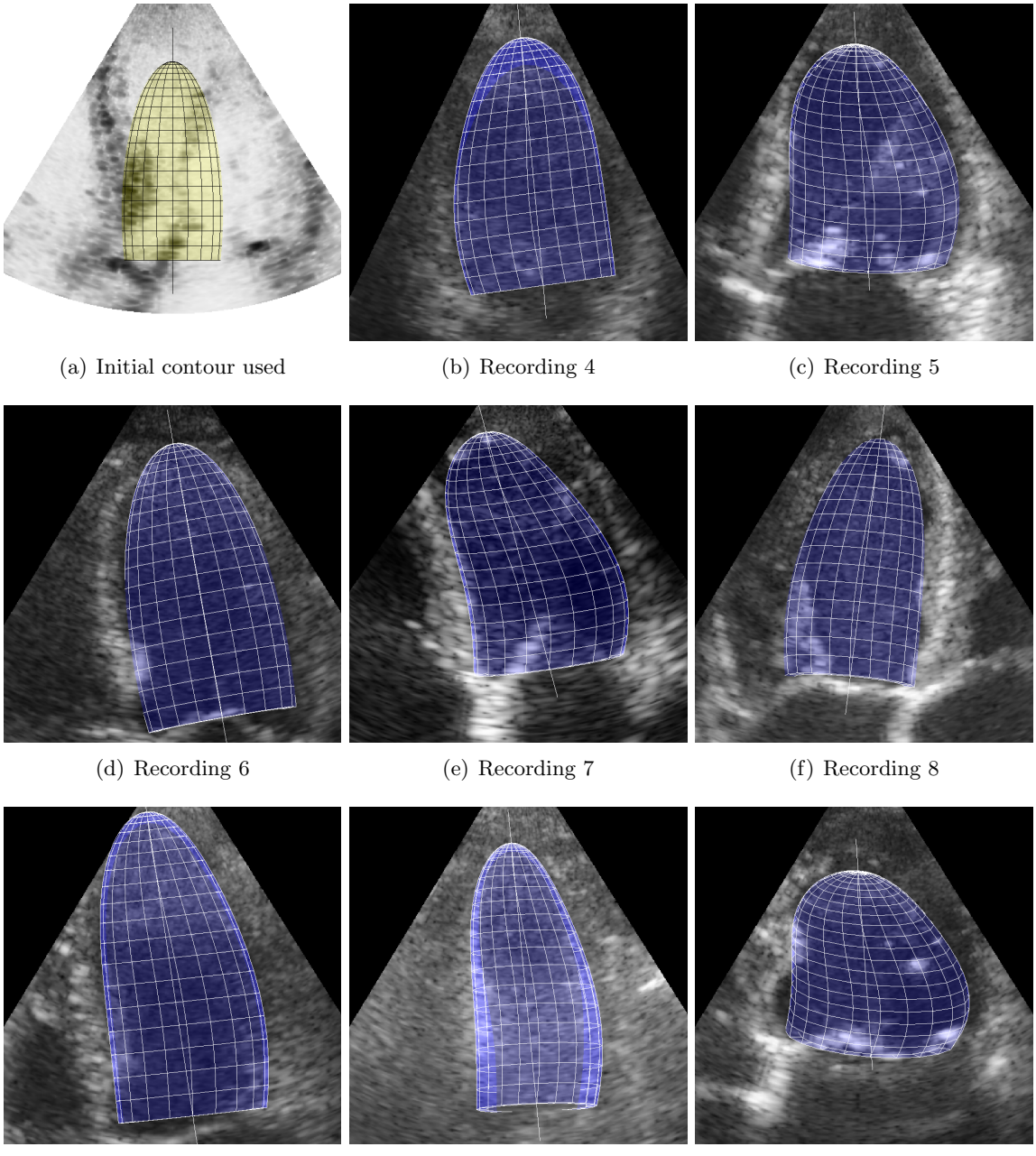
Table C.2: Results of automatic tracking within the GE Vingmed Ultrasound dataset (2/3).

Detailed tracking results		
Recording no	Tracking result	Comments
Recording 63	poor	4-chamber view with very poor contrast.
Recording 64	good	Poor image contrast and unclear walls.
Recording 65	good	Only tracks one wall.
Recording 66	good	Good tracking results.
Recording 67	adequate	Problems tracking mitral plane.
Recording 68	good	4-chamber view with one wall outside of the volume.
Recording 69	fair	4-chamber view of poor quality.
Recording 70	good	Good tracking.
Recording 71	poor	Unable to track apex.
Recording 72	good	Excellent tracking.
Recording 73	adequate	Has some problems following ventricle dynamics.
Recording 74	good	Good tracking.
Recording 75	good	Good tracking.
Recording 76	good	Good tracking.
Recording 77	good	Good tracking.
Recording 78	good	Problems reaching apex.
Recording 79	fair	Problems tracking apex and one wall.
Recording 80	adequate	Extreme MLA artifacts in image.
Recording 81	good	Excellent tracking.
Recording 82	good	Good tracking.
Recording 83	good	Excellent tracking in poor contrast.
Recording 84	good	4-chamber view.
Recording 85	fair	Unstable tracking results.
Recording 86	adequate	Only one wall visible.
Recording 87	good	Good tracking, but some model mismatch.
Recording 88	fair	Unstable mitral plane tracking, only one wall visible.
Recording 89	good	Good tracking, only one wall visible.
Recording 90	good	Excellent tracking.
Recording 91	good	Good tracking.
Recording 92	good	Good tracking.
Recording 93	good	Good tracking.

Table C.3: Results of automatic tracking within the GE Vingmed Ultrasound dataset (3/3).

Overall tracking results			
Quality	Count	Percentage	Comment
Good	52	59%	Automatic tracking of LV position and long-axis performed well.
Adequate	16	18%	Automatic tracking of LV position and long-axis performed with reduced accuracy.
Fair	14	16%	Automatic tracking of LV position and long-axis performed with low accuracy.
Poor	6	7%	Unable to automatically track of LV position and long-axis
n/a	5		Files missing or corrupt. Unable to perform tracking in them.

Table C.4: Overall result statistic from automatic tracking within the GE Vingmed Ultrasound dataset.

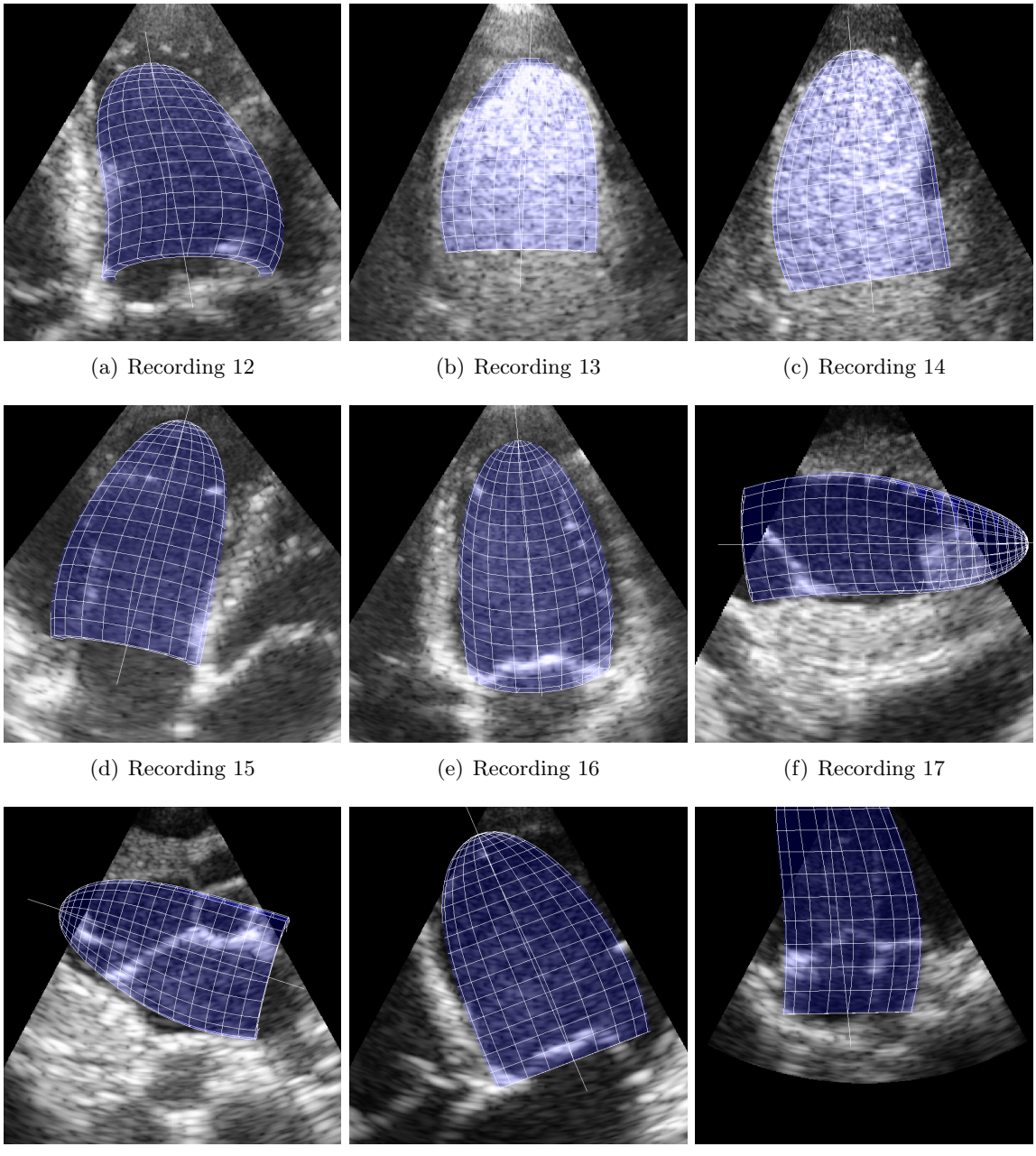


(a) Initial contour used (b) Recording 4 (c) Recording 5

(d) Recording 6 (e) Recording 7 (f) Recording 8

(g) Recording 9 (h) Recording 10 (i) Recording 11

Figure C.1: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (1/10).

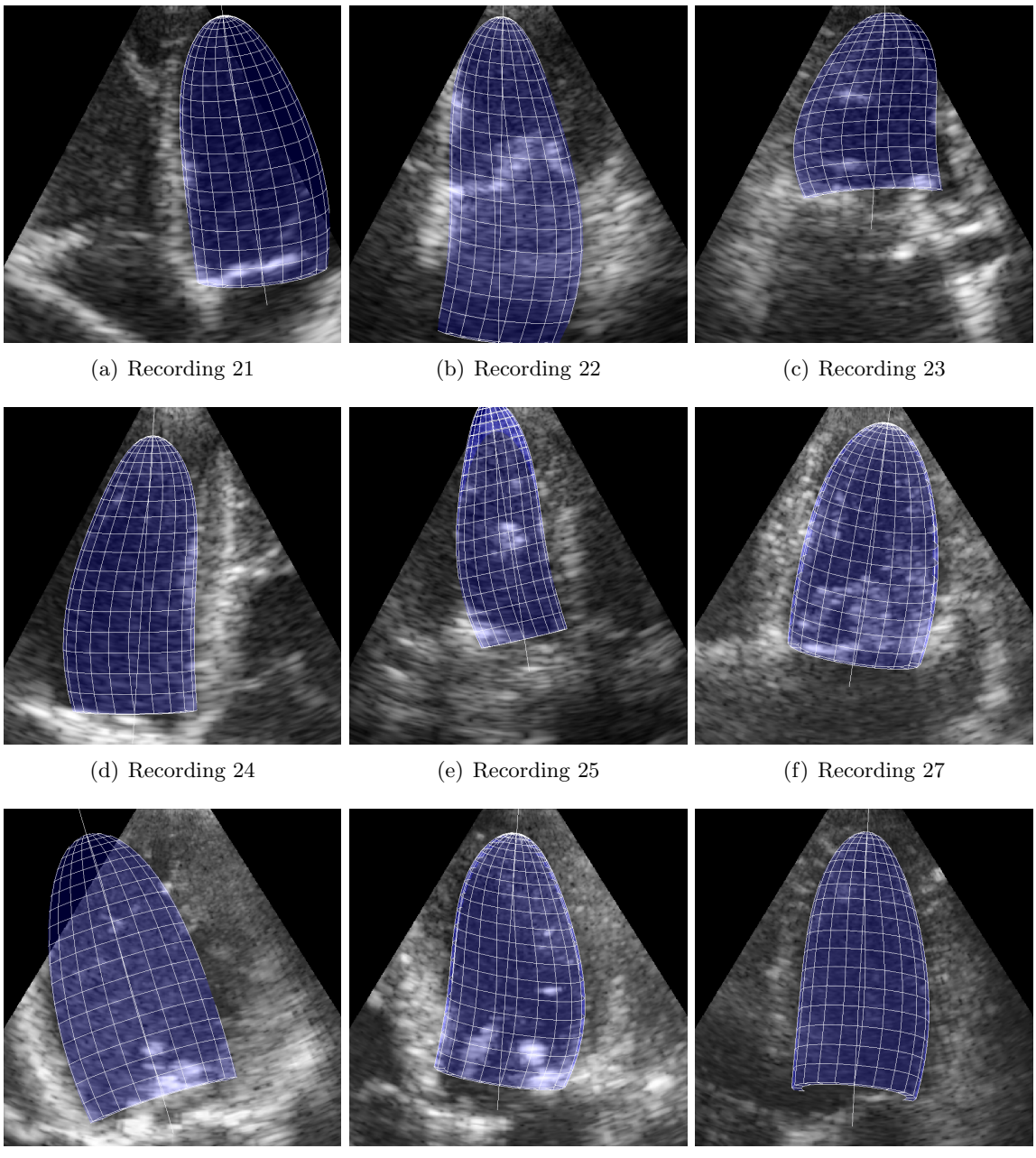


(a) Recording 12 (b) Recording 13 (c) Recording 14

(d) Recording 15 (e) Recording 16 (f) Recording 17

(g) Recording 18 (h) Recording 19 (i) Recording 20

Figure C.2: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (2/10).

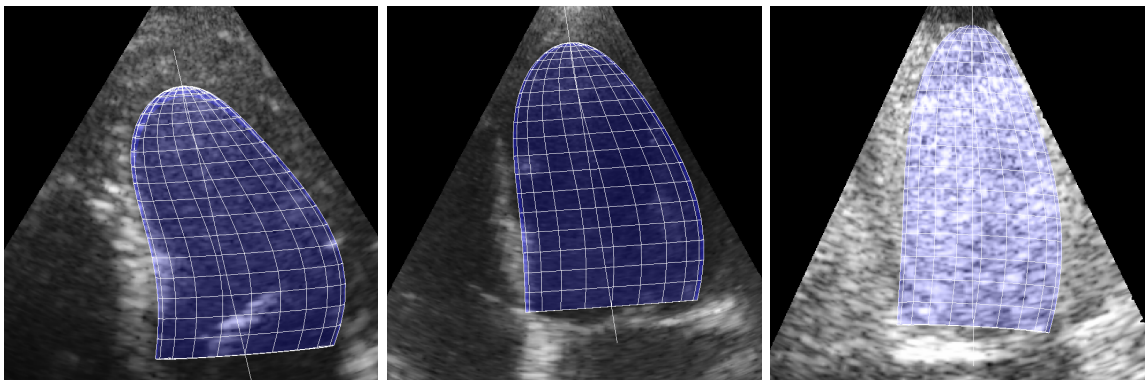


(a) Recording 21 (b) Recording 22 (c) Recording 23

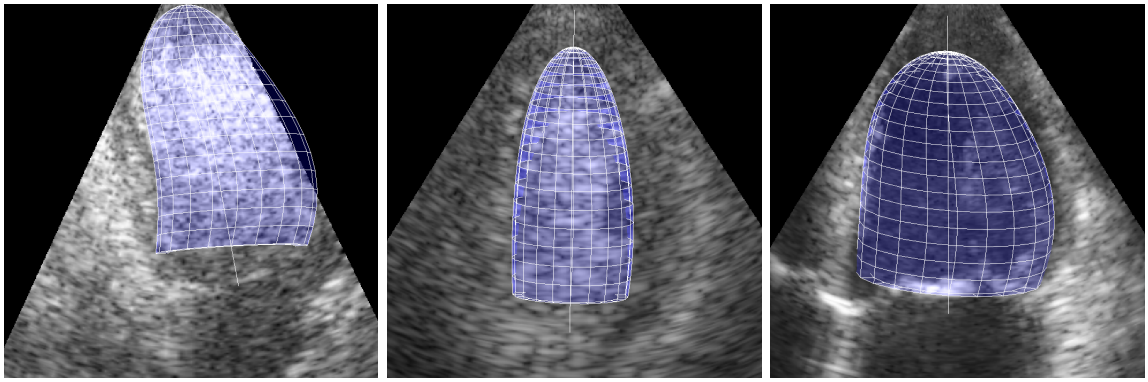
(d) Recording 24 (e) Recording 25 (f) Recording 27

(g) Recording 28 (h) Recording 30 (i) Recording 31

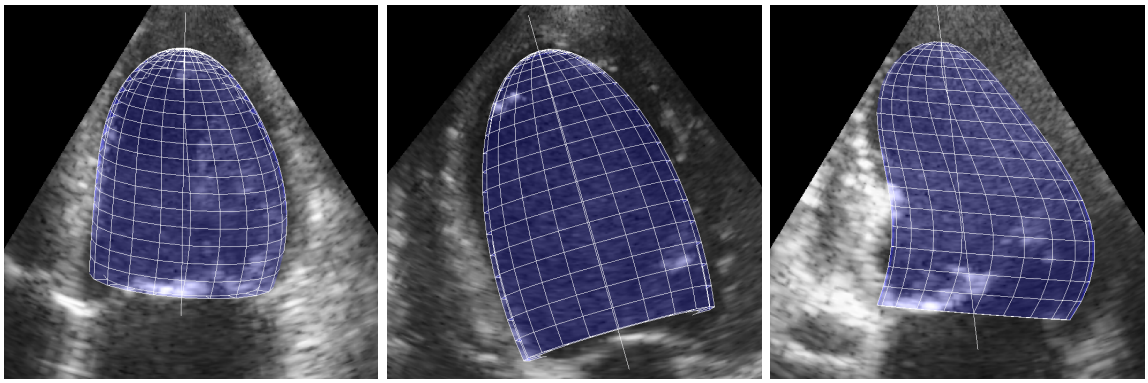
Figure C.3: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (3/10).



(a) Recording 32 (b) Recording 33 (c) Recording 34

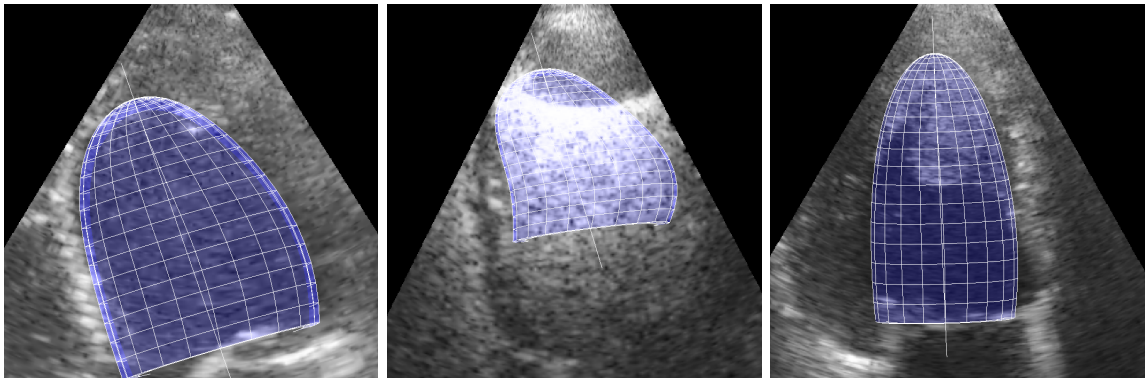


(d) Recording 35 (e) Recording 36 (f) Recording 37

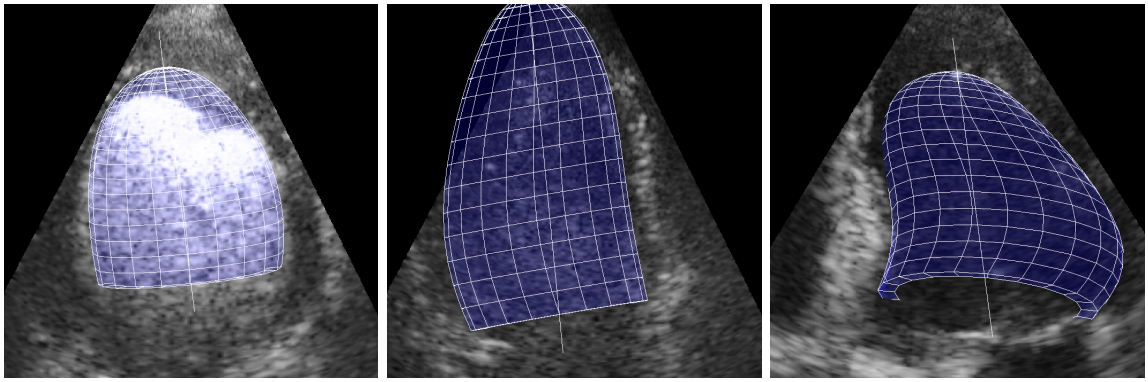


(g) Recording 38 (h) Recording 39 (i) Recording 40

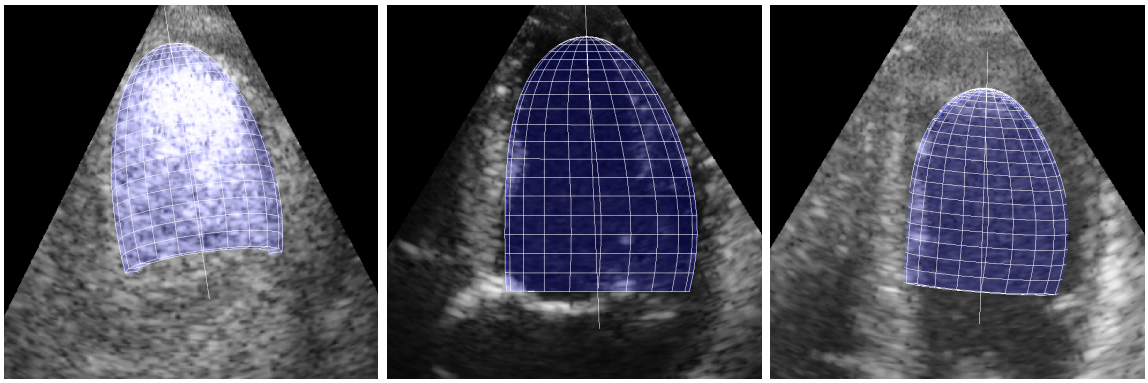
Figure C.4: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (4/10).



(a) Recording 41 (b) Recording 42 (c) Recording 43

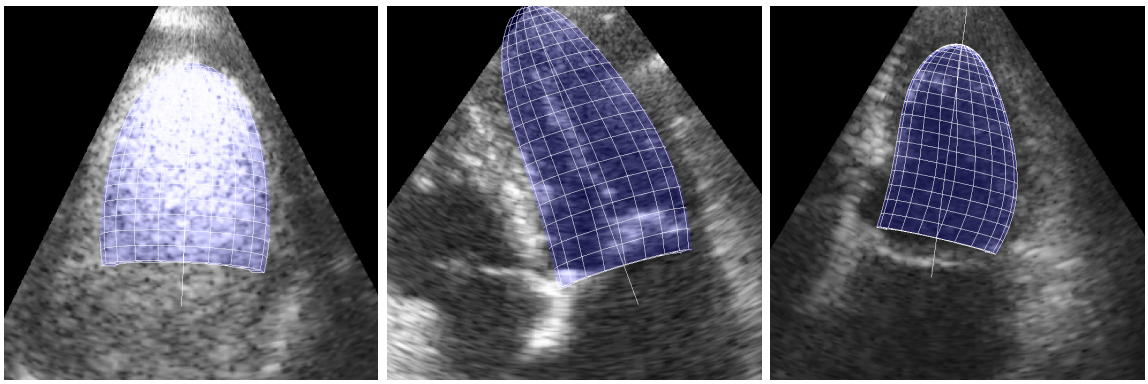


(d) Recording 44 (e) Recording 45 (f) Recording 46

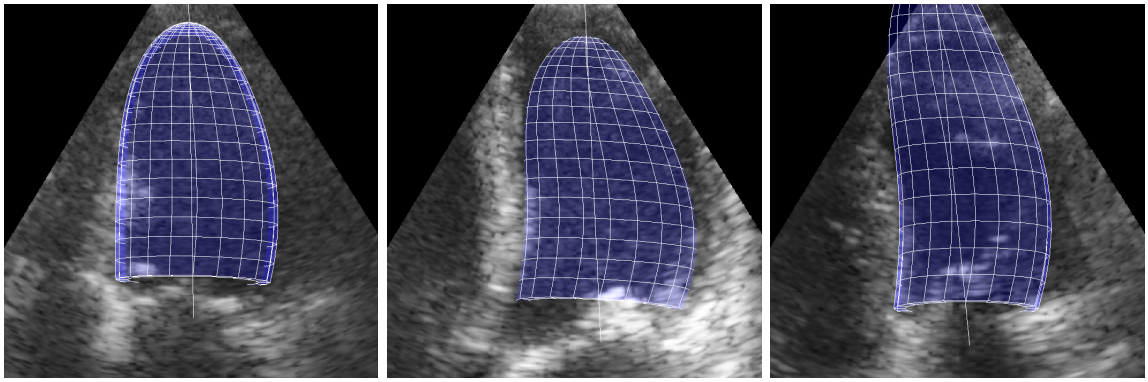


(g) Recording 47 (h) Recording 48 (i) Recording 49

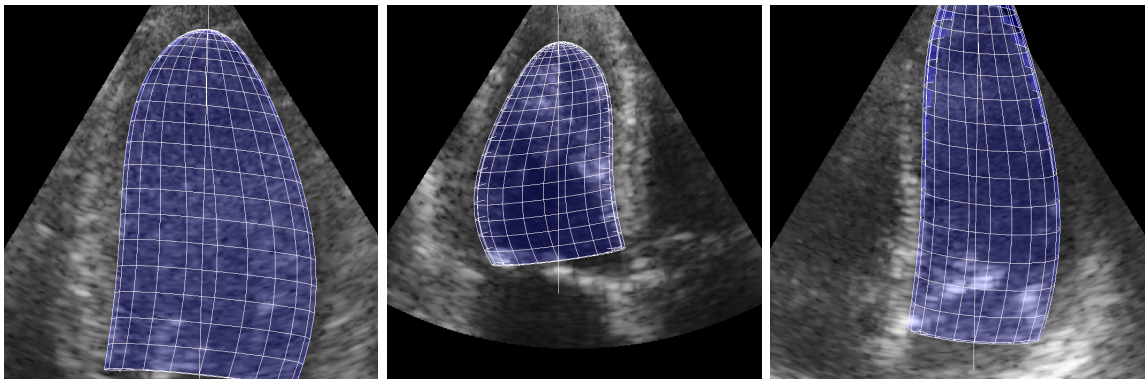
Figure C.5: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (5/10).



(a) Recording 50 (b) Recording 51 (c) Recording 52

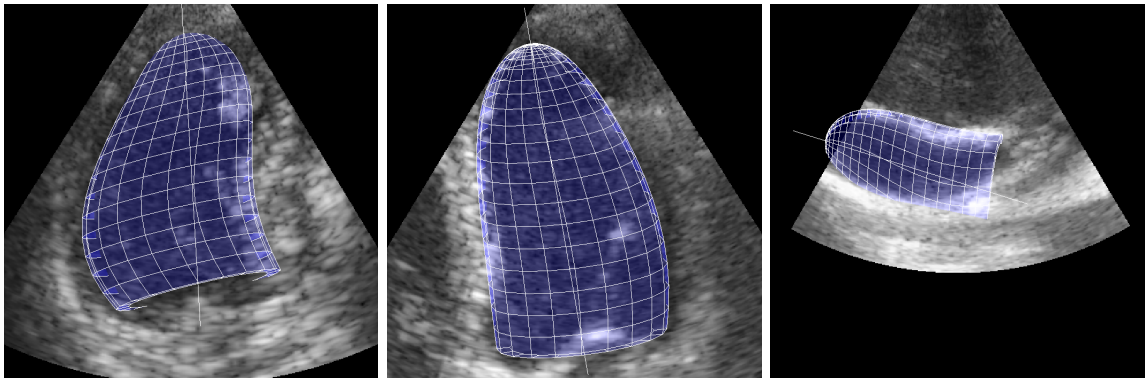


(d) Recording 53 (e) Recording 54 (f) Recording 55

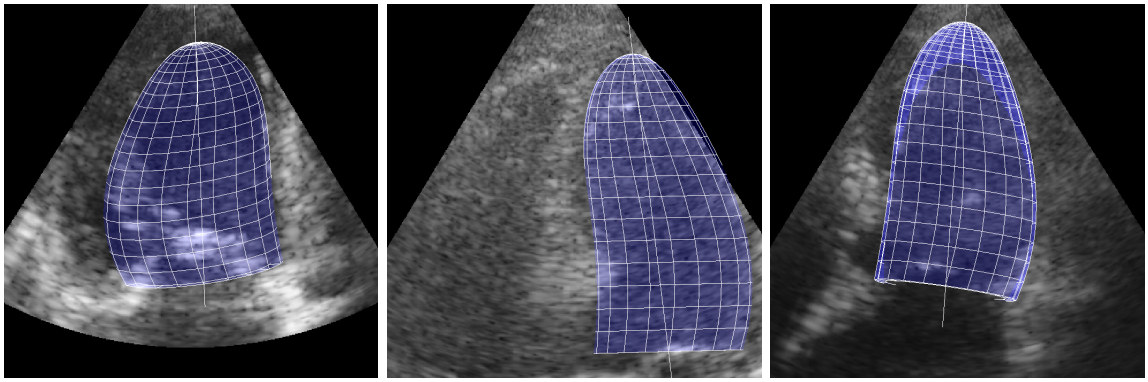


(g) Recording 56 (h) Recording 57 (i) Recording 58

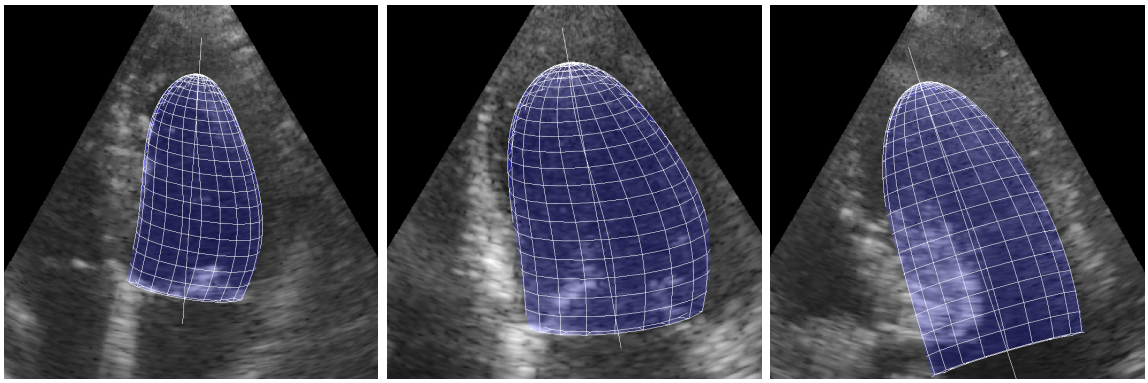
Figure C.6: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (6/10).



(a) Recording 59 (b) Recording 60 (c) Recording 61

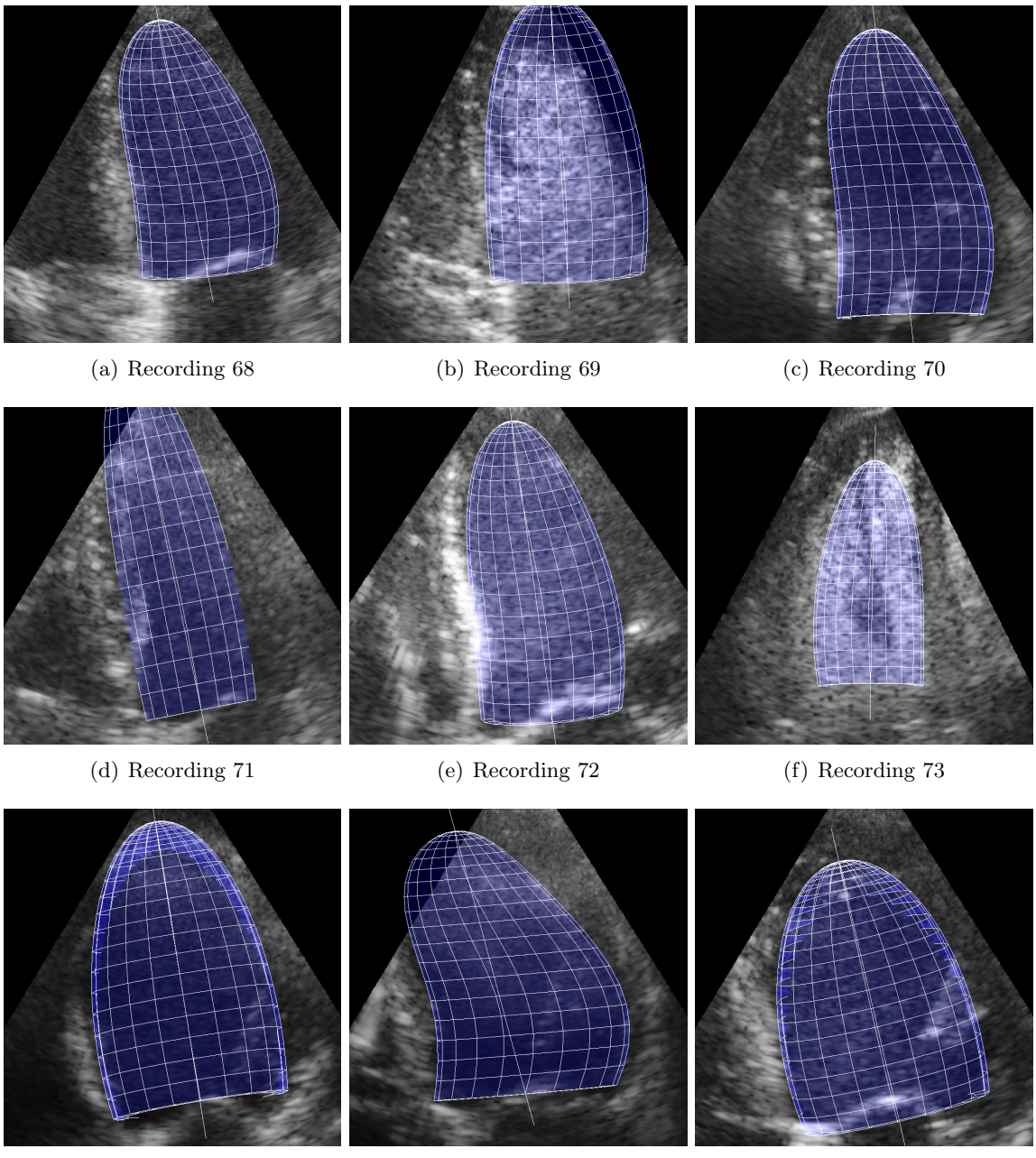


(d) Recording 62 (e) Recording 63 (f) Recording 64



(g) Recording 65 (h) Recording 66 (i) Recording 67

Figure C.7: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (7/10).

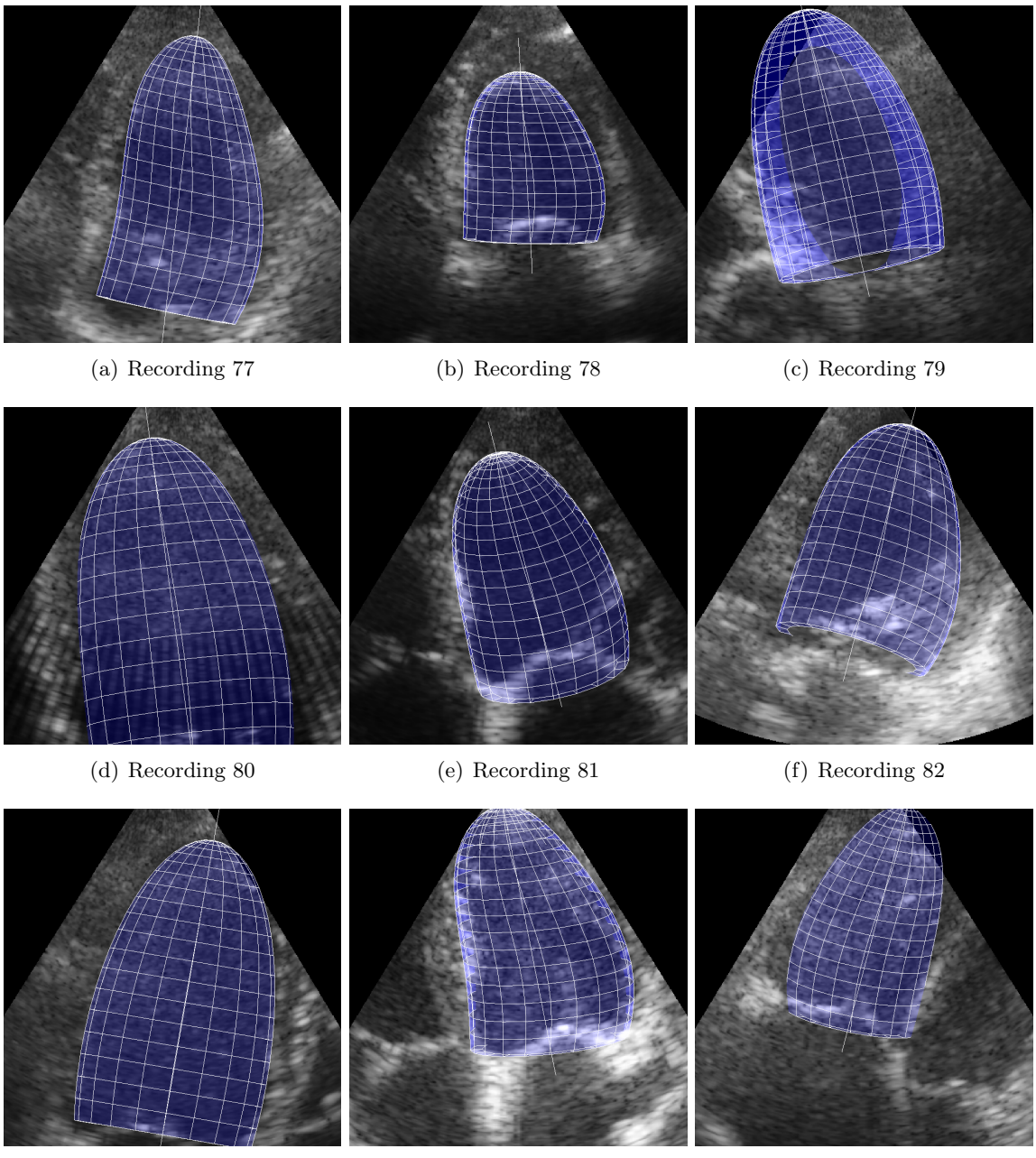


(a) Recording 68 (b) Recording 69 (c) Recording 70

(d) Recording 71 (e) Recording 72 (f) Recording 73

(g) Recording 74 (h) Recording 75 (i) Recording 76

Figure C.8: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (8/10).

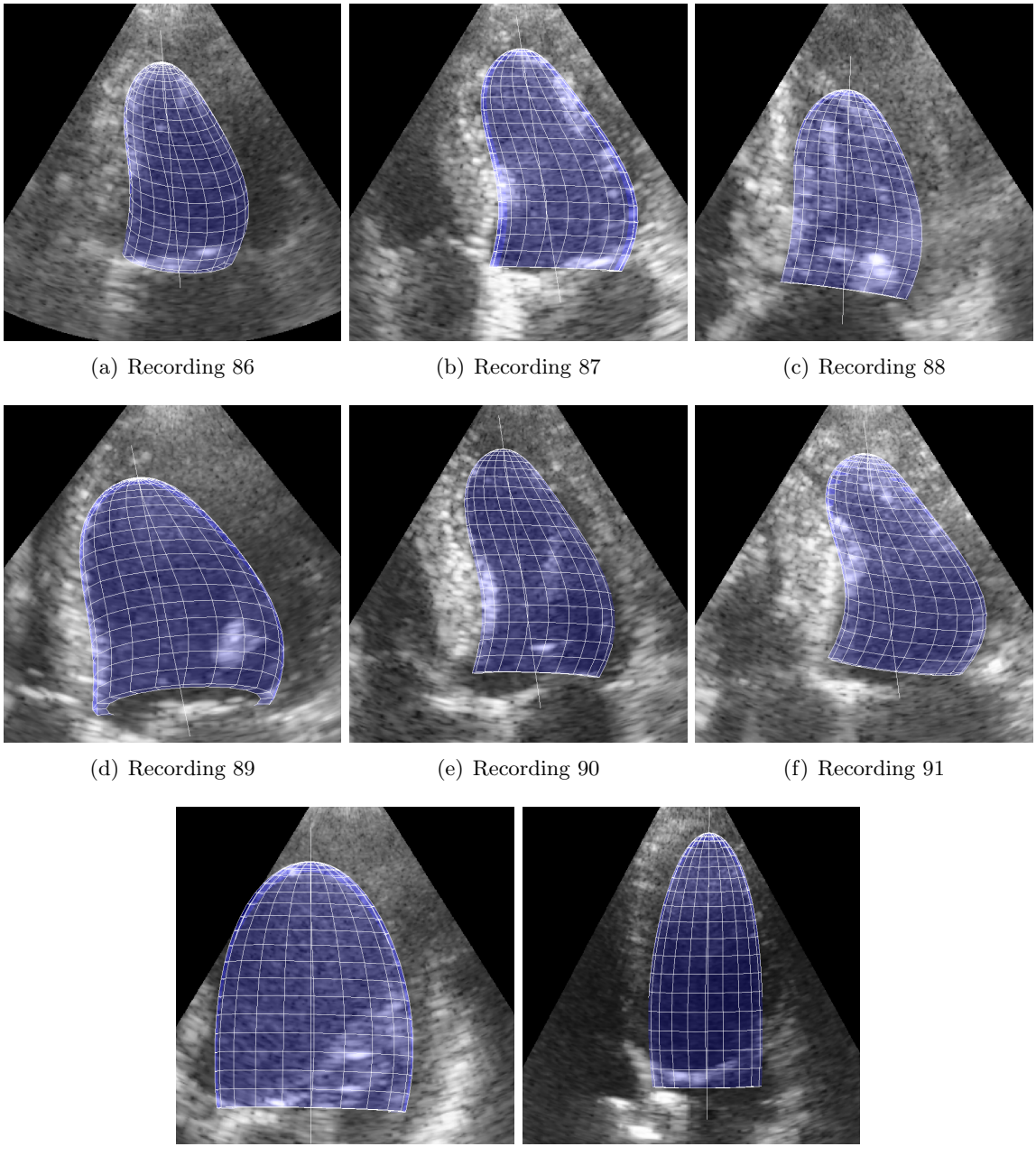


(a) Recording 77 (b) Recording 78 (c) Recording 79

(d) Recording 80 (e) Recording 81 (f) Recording 82

(g) Recording 83 (h) Recording 84 (i) Recording 85

Figure C.9: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (9/10).



(a) Recording 86 (b) Recording 87 (c) Recording 88

(d) Recording 89 (e) Recording 90 (f) Recording 91

(g) Recording 92 (h) Recording 93

Figure C.10: Screenshots from automatic tracking within the GE Vingmed Ultrasound dataset (10/10).

Bibliography

[1] Elsa D. Angelini, Andrew F. Laine, Shin Takuma, Jeffrey W. Holmes, and Shunichi Homma. LV volume quantification via spatiotemporal analysis of real-time 3-d echocardiography. *IEEE Transactions on Medical Imaging*, 20(6):457–469, 2001.

[2] S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, February 2002.

[3] Y. Bar-Shalom. *Tracking and data association.* Academic Press Professional, Inc., San Diego, CA, USA, 1987.

[4] Yaakov Bar-Shalom, Thiagalingam Kirubarajan, and X.-Rong Li. *Estimation with Applications to Tracking and Navigation.* John Wiley & Sons, Inc., New York, NY, USA, 2002.

[5] Alan H. Barr. Global and local deformations of solid primitives. In *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*, pages 21–30, New York, NY, USA, 1984. ACM Press.

[6] Andrew Blake, Rupert Curwen, and Andrew Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145, 1993.

[7] Andrew Blake and M. Isard. *Active Contours: The Application of Techniques from Graphics,Vision,Control Theory and Statistics to Visual Tracking of Shapes in Motion.* Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1998.

[8] Andrew Blake, Michael Isard, and David Reynard. Learning to track the visual motion of contours. *Artificial Intelligence*, 78(1-2):179–212, 1995.

[9] Robert Grover Brown and Patrick Y. C. Hwang. *Introduction to Random Signals and Applied Kalman Filtering, 3rd Edition.* Prentice Hall, 1996.

[10] Chi-Tsong Chen. *Linear System Theory and Design, third edition.* Oxford University Press, 1999.

[11] Dorin Comaniciu, Xiang Sean Zhou, and Sriram Krishnan. Robust real-time myocardial border tracking for echocardiography: an information fusion approach. *IEEE Transactions on Medical Imaging*, 23(7):849–860, 2004.

[12] Arthur Gelb. *Applied Optimal Estimation.* The MIT Press, 1974.

[13] Michael Isard and Andrew Blake. Condensation - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1):5–28, August 1998.

[14] G. Jacob, J. A. Noble, M. Mulet-Parada, and A. Blake. Evaluating a robust contour tracker on echocardiographic sequences. *Medical Image Analysis*, 3(1):63–75, 1999.

[15] Gary Jacob, J. Alison Noble, Christian P. Behrenbruch, Andrew D. Kelion, and Adrian P. Banning. A shape-space based approach to tracking myocardial borders and quantifying regional left ventricular function applied in echocardiography. *IEEE Transactions on Medical Imaging*, 21(3):226–238, 2002.

[16] Gary Jacob, J. Alison Noble, Andrew D. Kelion, and Adrian P. Banning. Quantitative regional analysis of myocardial wall motion. *Ultrasound in Medicine & Biology*, 27(6):773–784, 2001.

[17] Andrew H. Jazwinski. *Stochastic Processes and Filtering Theory*. Academic Press, 1970.

[18] Guo Luo and Pheng Heng. LV shape and motion: B-spline-based deformable model and sequential motion decomposition. *IEEE Transactions on Information Technology in Biomedicine*, 9(3):430–446, 2005.

[19] Sotiris Malassiotis and Michael G. Strintzis. Tracking the left ventricle in echocardiographic images by learning heart dynamics. *IEEE Transactions on Medical Imaging*, 18(3):282–290, 1999.

[20] Thomas B. Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231–268, 2001.

[21] J. Park, D. Metaxas, A. Young, and L. Axel. Deformable models with parameter functions for cardiac motion analysis from tagged mri data. *IEEE Transactions on Medical Imaging*, 15(3):290–298, June 1996.

[22] Natan Peterfreund. Robust tracking of position and velocity with kalman snakes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(6):564–569, 1999.

[23] Stein Inge Rabben, Anders Herman Torp, Asbjørn Støylen, Stig Slørdahl, Knut Bjørnstad, Bjørn Olav Haugen, and Bjørn Angelsen. Semiautomatic contour detection in ultrasound M-mode images. *Ultrasound in Medicine & Biology*, 26(2):287–296, 2000.

[24] Espen W. Remme, Alistair A. Young, Kevin F. Augenstein, Brett Cowan, and Peter J. Hunter. Extraction and quantification of left ventricular deformation modes. *IEEE Trans. Biomedical Engineering*, 51(11):1923–1931, 2004.

[25] Charles W. Therrien. *Discrete Random Signals and Statistical Signal Processing*. Prentice Hall, 1992.

[26] Wikipedia. Cardiac cycle — `http://en.wikipedia.org/wiki/Cardiac_cycle`. [Online; accessed 11-June-2006].

[27] Wikipedia. Contrast enhanced ultrasound — `http://en.wikipedia.org/wiki/Contrast_enhanced_ultrasound`. [Online; accessed 11-June-2006].

[28] Wikipedia. Echocardiography — http://en.wikipedia.org/wiki/Echocardiography. [Online; accessed 11-June-2006].

[29] Wikipedia. Ejection fraction — http://en.wikipedia.org/wiki/Ejection_fraction. [Online; accessed 11-June-2006].

[30] Wikipedia. Heart — http://en.wikipedia.org/wiki/Heart. [Online; accessed 11-June-2006].

[31] Wikipedia. Linear interpolation — http://en.wikipedia.org/wiki/Linear_interpolation. [Online; accessed 11-June-2006].

[32] Wikipedia. Medical ultrasonography — http://en.wikipedia.org/wiki/Medical_ultrasonography. [Online; accessed 11-June-2006].

[33] Wikipedia. Phased array — http://en.wikipedia.org/wiki/Phased_array. [Online; accessed 11-June-2006].

[34] Wikipedia. Polar coordinate system — http://en.wikipedia.org/wiki/Polar_coordinate_system. [Online; accessed 11-June-2006].

[35] V. Zagrodsky, V. Walimbe, C.R. Castro-Pareja, J.X. Qin, J.M. Song, and R. Shekhar. Registration-assisted segmentation of real-time 3-d echocardiographic data using deformable models. *IEEE Transactions on Medical Imaging*, 24(9):1089–1099, September 2005.

[36] Xiang Sean Zhou, Dorin Comaniciu, and Alok Gupta. An information fusion framework for robust shape tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(1):115–129, 2005.