NTNU

Norwegian University of
Science and Technology

Practical use of Block-Matching in 3D Speckle Tracking

Karl Espen Nielsen

Master of Science in Computer Science

Submission date:	June 2009
Supervisor:	Bjørn Olstad, IDI
Co-supervisor:	Hans Torp, ISB
	Gabriel Kiss, ISB
	Fredrik Orderud, IDI

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

Problem Description

3D speckle tracking is used to track material points between successive image frames, and can be used to estimate material strain or regional contraction of the heart. Block-matching algorithms are typically used to track the speckle patterns between frames.

The purpose of this thesis is to study methods for improving the results from the block-matching. The methods should be tested with 3D ultrasound recordings of the left ventricle and be compared and evaluated by their ability track contours in real-time tracking.

Assignment given: 15. January 2009
Supervisor: Bjørn Olstad, IDI

Abstract

In this thesis, optimizations for speckle tracking are integrated into an existing framework for real-time tracking of deformable subdivision surfaces. This is employed in the segmentation of the the left ventricle (LV) in 3D echocardiography. The main purpose of the project was to optimize the efficiency of material point tracking, this leading to a more robust LV myocardial deformation field estimation.

Block-matching is the most time consuming part of speckle tracking, and the corresponding algorithms used in this thesis are optimized based on a Single Instruction Multiple Data (SIMD) model, in order to achieve data level parallelism. The SIMD model is implemented by using Streaming SIMD Extensions (SSE) to improve the processing time for the computation of the sum of absolute differences, one possible metric for block matching purposes.

Furthermore, a study is conducted to optimize parameters associated with speckle tracking in regards to both accuracy and computation time. This is tested by using simulated data sets of infarcted ventricles in 3D echocardiography. More specifically, the tests examine how the size of kernel blocks and search windows affect the accuracy and processing time of the tracking. It also compares the performance of kernel blocks specified in cartesian and beamspace coordinates. Finally, tracking-accuracy is compared and measured in different regions (apical, mid-level and basal segments) of the LV.

Preface

This thesis has been written in the 10th semester of the Master of Technology program at the Department of Computer and Information Science (IDI) at the Norwegian University of Science and Technology (NTNU), during the spring of 2009.

I would especially like to thank my supervisor Gabriel Kiss, for very helpful guidance and feedback. I will also like to thank Professor Hans Torp at the Department of Circulation and Medical Imaging, and Fredrik Orderud and Professor Bjørn Olstad at the Department of Computer and Information Science for useful input and feedback.

Trondheim, 11 June 2009

Karl Espen Nielsen

Contents

Chapter 1

Introduction

Cardiovascular diseases are among the most common causes of death in Norway and a large number of these deaths are caused by ischemic heart diseases. Even though deaths due to ischemic heart diseases have become less common in the last years, they are still responsible for 14% of all deaths in Norway [26].

The regional function of the left ventricle is known to be of high diagnostic value for ischemic heart diseases. Speckle tracking in 2D ultrasound images has been shown as a useful tool for assessing this function [12, 17]. With the recent developments in ultrasound systems there are now possible to record full volume data in real-time, and thus makes it possible to perform speckle tracking in three dimensions [6].

The most common methods for performing speckle tracking are algorithms based on optical flow or block matching. They are both computational costly methods. However, by employing a Single Instruction Multiple Data (SIMD) model to the block matching, a significant increase in performance can be achieved due to data level parallelism.

This thesis has two main goals:

The first goal of this thesis is to integrate optimizations for block-matching into an existing framework for fully automatic tracking of deformable structures called: Real-time Contour Tracking Library (RCTL) [20, 21, 22]. The implementation used is based on the work performed during the Specialization Project in Complex Computer Systems, fall 2008 [19]. The optimizations use Streaming SIMD Extensions (SSE) to reduce processing time of computation of the sum of absolute differences (SAD).

The second goal is to look into how block matching can be used most efficiently in speckle tracking of the left ventricle. This thesis present results on how well block matching performs with speckle tracking in respect to both accuracy and processing time depending on acquisition parameters. The measurements of the tracking are performed on simulated 3D echocardiographic recordings of the left ventricle. The impact the size of the kernel block and search window have on the results are investigated. The tracking accuracy is evaluated for various segments of the ventricle. The thesis also compares the performance of block matching with kernel blocks specified in cartesian and beamspace coordinates.

Chapter 2

Background

2.1 Heart Physiology

The heart is a muscular organ. It has a conical blunt shape and is approximately the size of a closed fist [24]. The rounded point of the cone is called the apex and the large flat part at the opposite end is called the base. The apex of the heart is situated at the leftmost point at the downward side of the heart [11]. The heart is located in the thoracic cavity. It is between the lungs and slightly to the left of the breastbone.

The walls of the heart are called myocardium and are composed by a special type of muscular tissue. A coordinated contraction of the muscle cells in the myocardium propel the blood out of the ventricles and into the blood vessels in the circulatory system. This stage of the cardiac cycle is called the systole. This contraction is followed by a phase called diastole, where the heart relaxes, and the atria and ventricles are filled with blood.

If the blood supply to a part the heart is interrupted, cells in the myocardium will start to die. This is called a myocardial infraction. When the cells in an area dies a scar tissue will be formed in their place. This scar tissue reduces the ventricular function by inhibiting contractions in the affected areas.

2.1.1 Left Ventricle

The left ventricle is one of the four chambers in the heart. The placement can be shown in Figure 2.1. It receives oxygenated blood and pumps it out

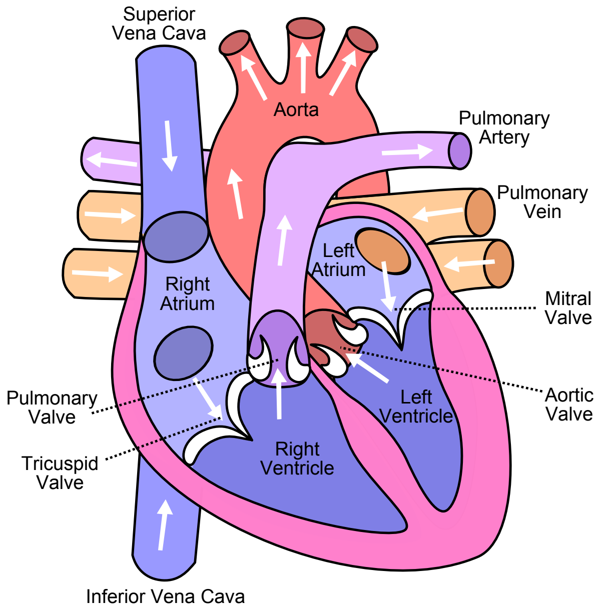


Figure 2.1: The human heart. The left ventricle is shown at the right side in the figure. Figure from [27].

via the aorta to the blood vessels in the circulatory system.

The left ventricle has a conical shape and is therefore usually segmented into the following regions when addressing different vertical sections of the myocardium [4]:

- Apex
- Apical area
- Mid-level area
- Basal area

The basal area is the area around the base of the conical shape of the ventricle and the apical area is the area close to the apex of the cone, as shown in Figure 2.2.

The pumping function of the ventricle is caused by a combination of contractions in the myocardium and a torsion of the ventricle. This torsion is generated by oppositely directed apical and basal rotations.

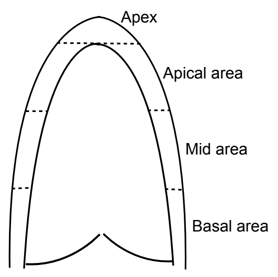


Figure 2.2: Long axis view of the vertical segmentation of the left ventricle.

2.2 Ultrasound

Ultrasound is the term for acoustic waves with frequencies above the upper limit for human hearing. All acoustic frequencies above 20 kHz are classified as ultrasound, but ultrasound in medical diagnostics usually have frequencies in the range between 2 MHz and 10 MHz [2]. Ultrasound imaging is based on sending an ultrasound pulse into a medium and listening for echoes generated from changes in the structure of the carrier medium.

Different types of tissue have different acoustic impedance. When a sound wave hits a transition between two layers of tissue some of the wave signal will be reflected. How much of the signal is reflected depends on the difference in acoustic impedance between the two layers. A large difference will generate a strong reflection and a small difference will generate a weak reflection. Reflections of acoustic waves are called echoes. When listening for these echoes and by timing how long it takes before they return, one can estimate how far away structural changes in the tissue are located. The sound travels with different speeds in different media, as shown in Table 2.1. In air it is 330 m/s and in water it is about 1500 m/s. Soft tissues have similar acoustic properties as water. The speed of sound here is about 1540 m/s and vary little between different tissues. Since we know the speed of the sound, the distance to tissue changes are given by the following equation: $r = \frac{tc}{2}$. Where t is time from the pulse was sent to the echo is received and c is the speed of sound. The equation is divided by a factor of 2 because the wave has to travel the distance two times.

The ultrasound waves are produced by a transducer. When an element in a transducer has sent a wave the received signal can be used to produce an one-dimensional image of a line into the medium. An ultrasound transducer

Material	Mass density kg/m^3	Sound velocity m/s	Acoustic impedance $10^6 kg/(m^2 s)$
Fat	950	1440	1.37
Blood	1025	1570	1.61
Muscles	1070	1542 - 1626	1.65 - 1.74
Air	1.2	330	0.0004
Salt Water	1025	1531	1.569

Table 2.1: List of properties for different materials [2].

contains an array of elements that alternates between sending and receiving ultrasound signals. By having slightly different angles for each element in the transducer, an image of a two-dimensional slice of the medium is acquired. To produce 3D images a two-dimensional array of elements is required in the transducer. This is illustrated in Figure 2.3.

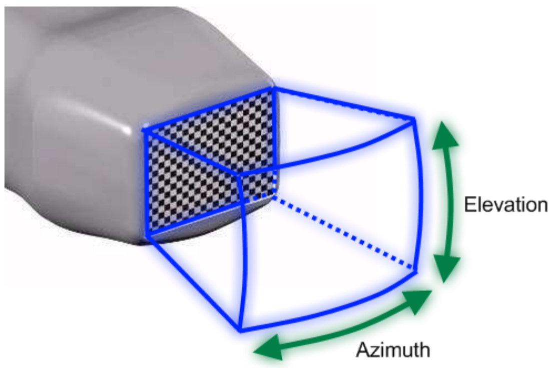


Figure 2.3: An ultrasound transducer used in 3D imaging. The layout of the matrix array of elements is illustrated. The orientation of the azimuth and elevation direction is indicated. The direction along the beam is called depth or range direction. Figure adapted from [14].

3D recordings are usually stored in a coordinate system called *beamspace*. This coordinate system is similar to polar coordinates, with the probe representing the origin. The length along the beams is the radial or *depth* direction. The direction across a row of beams is called the *azimuth* direction and the direction across the different rows of beams or planes is called the *elevation* direction.

There is usually the axis along the depth direction that has the highest resolution. The azimuth and elevation axis require one transducer element for each data point, which usually makes it more convenient to have the resolution significantly lower along these axes. While the density of the resolution along the depth axis is constant, the density of the data elements along the other two axes is dependent of the distance from the probe. This provides extra challenges when processing the image data.

Ultrasound does not produce as good images as computed tomography (CT) or magnetic resonance imaging (MRI). However, the equipment for ultrasound are less expensive and more portable than the equipment for the other methods. Ultrasound is also capable of producing real-time 3D images while the other methods require several minutes to perform a 3D scan. The portability and simplicity of ultrasound sets other expectations to the processing of the results. It is desired to produce the results real-time while recording, compared to CT and MRI where one usually have several hours for processing the image data.

2.2.1 Echocardiography

Echocardiography is medical ultrasound used to diagnose cardiovascular diseases. It is widely used for producing 2D images of slices of the heart. During the last years have also 3D ultrasound been introduced for imaging of the heart.

Unfortunately, the heart has a difficult position for ultrasound probes. It is partially hidden by the left lung, and it is situated inside the rib cage. The simplest and most common way of recording the images is transthoracic (through the chest wall). The probes for this purpose have to record the images through the small spaces between the the ribs, as the bones reflect almost all the signal due to high acoustic impedance. This sets restrictions for the size of the probe and a smaller probe means less elements for the recording of the data, which results in limited spatial resolution of the images. 3D echographic images usually have lower resolution along the elevation axis than the azimuth axis.

Echocardiography has several uses in diagnostics, one of them is its ability to discover myocardial infarctions. This thesis focus on how ultrasound can be used to assess the regional function of the left ventricle to diagnose myocardial infarctions.

2.3 Speckle Tracking

Speckle patterns are a characteristic feature of ultrasound imaging. Since no medium in the body is completely flat and homogeneous, small imperfections and structures in the tissue scatter the waves and produce interference patterns. This type of pattern is called a speckle pattern.

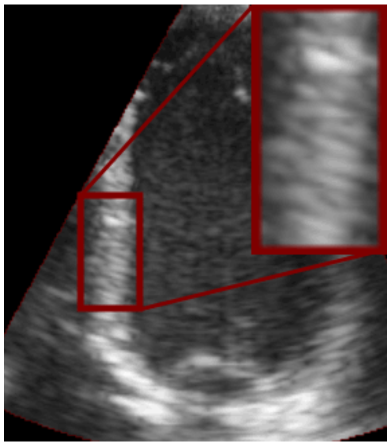


Figure 2.4: A slice from an ultrasound image of the left ventricle showing the speckle pattern in the myocardium.

In ultrasound imaging waves sent from each element in the transducer will interfere with each other when they hit different structures. This will produce a speckle pattern in the resulting image. These patterns are deterministic, but they are not correlated to the structures in the image [1]. Because the speckle pattern not corresponds to its underlying structure, it has a negative effect on the quality of the image. There have been shown that speckle greatly reduce the ability to detect features in the image [3]. Since the speckle pattern is deterministic it also has the property that its movements follow the movement of the underlying tissue [18]. Deformations in the myocardium may therefore be tracked by tracking the movement of the corresponding speckle pattern.

Speckle tracking is a common technique for tracking how structures move over time in echocardiography. When a region from a frame of an ultrasound recording is selected it will contain a speckle pattern. Speckle tracking consists of detecting how this pattern moves from frame to frame in a recording, and thereby determining the movements of the tissues in the same area.

Common techniques used for speckle tracking are block matching algorithms or differential based optical flow algorithms.

An advantage with speckle tracking in comparison to traditional edge detection is that it not only finds the change in shape between frames, but it tracks the actual movements of the material points. Certain motions like strain and torsion are assessed more accurately with speckle tracking, because a large motion in the structure may result in only small changes in the shape. In echocardiography these motions are interesting to observe as they provide important information of the regional myocardial function.

Speckle tracking has become a widely used and clinically proven technique for assessing strain in 2D recordings [12, 17]. However, the limited processing capacity available have been an issue for recordings of volumetric data, but in the last few years studies have shown promising results for three-dimensional speckle tracking [21, 5].

2.4 Block-Matching

Block-matching is used for measuring how two similar images or segments of images are on pixel level [25].

Block-matching is a common method for motion estimation in digital recordings. The method is used both in video compression and in applications for visual tracking such as speckle tracking.

When performing block-matching, a block is selected from a frame i and matched with regions at various offsets in an adjacent frame j. Frame j may be either before or after frame i, but it is usually the following frame. The block is often referred to as the kernel and the set of regions it is matched against in frame j is called the search window. At each offset a metric is used to evaluate how well the block fits at the respective position. There are several metrics for evaluating this fitness value.

The algorithms for searching for the best of can be divided in two groups: full-search algorithms and fast-search algorithms. Full-search algorithms are algorithms that evaluate the kernel for every offset in the search window. The fast-search algorithms use heuristics to only evaluate a subset of the possible offsets. While fast-search algorithms are significantly faster than the full-search algorithms, they usually perform worse at finding accurate matches.

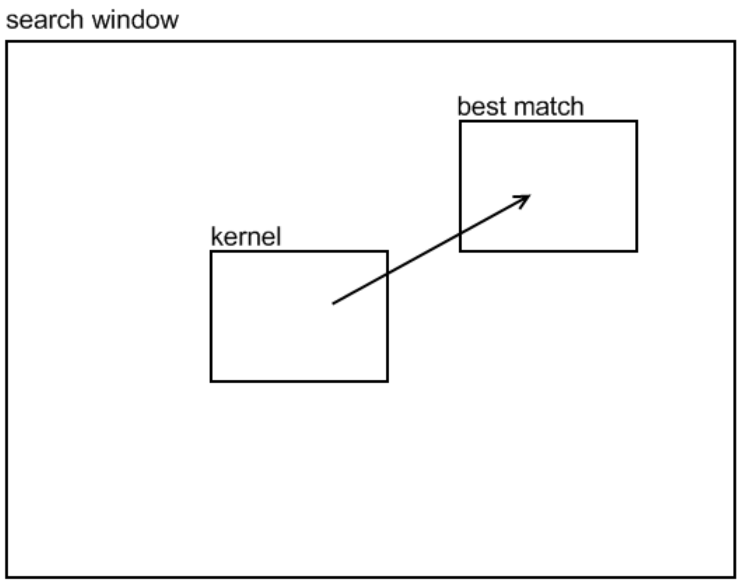


Figure 2.5: Block-matching algorithms search for the best match for a kernel block within a given search window.

2.4.1 Matching metrics

Sum of absolute differences

Sum of absolute differences (SAD) is an evaluation metric for block-matching. It computes the absolute value of the difference between each point in a kernel block and the corresponding pixels in a reference block. This function shows how SAD is computed in 2D block matching when the block size is N*N samples and C_{ij},R_{ij} are samples from respectively the current area and the reference area [23]. The lowest SAD value indicates the best match.

$$SAD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} |C_{ij} - R_{ij}|$$

SAD is the simplest and fastest evaluation metric for block-matching that takes every pixel in the block into account. Because SAD analyze each point separately, it is well suited for parallelization.

Sum of squared differences

Sum of squared differences (SSD) is similar to SAD. Instead of using the absolute value of the difference in each point, it computes the square of the value. This gives a higher penalty for elements with a large distance. The method is slightly more computationally demanding than SAD, but it often provides better results.

$$SSD = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} (C_{ij} - R_{ij})^2$$

Normalized cross-correlation

Normalized cross-correlation is a standard method for estimating how correlated two data sets are.

First the data are normalized. This is that the mean is subtracted and the result is divided by the standard deviation for each value in the data sets. This function shows how the cross-correlation is computed, where k is the kernel block and s is a sub-set of the search space. A higher value of the cross-correlation indicate a better match.

$$Cross - correlation = \sum_{x,y} \frac{(k(x,y) - u_k)\,(s(x,y) - u_s)}{\sigma_k\,\sigma_s}$$

The computation of cross-correlation requires more time than both SAD and SSD.

2.5 SIMD Intstructions

Single instruction, multiple data (SIMD) is one of four classifications of computer architectures proposed by M.J. Flynn in 1966 [9, 7]. The traditional method for computer architecture was to have a single processing unit with a single instruction set that computes one instruction at a time. Flynn called this: Single Instruction, Single Data stream (SISD). SIMD architectures still have only one set of instructions, but they are able to execute one type of instruction simultaneously on multiple data blocks.

The SIMD model was first implemented on large supercomputers. These were computers that were typically used for adding together large arrays of data. Later SIMD architecture was implemented in smaller computers. Most of computers today have support for SIMD instructions of some degree.

2.5.1 Streaming SIMD Extensions

Streaming SIMD Extensions (SSE) is an extension first developed by Intel on their x86 architecture, it has later also been implemented in AMD-processors. This extension utilizes SIMD at a small scale [15]. In addition to the standard 32-bit registers x86 computers are equipped with separate 128-bit registers. These registers are often called floating-point registers because they are mainly used for floating point operations, but they are also used by SSE instructions. SSE enables processing of these registers as if they contained an array of smaller values. SSE instructions may regard the registers as containing:

- One 128-bit value
- Two 64-bit values
- Four 32-bit values
- Eight 16-bit values
- Sixteen 8-bit values

The instructions in SSE enable the processing unit to execute operations on all values in a 128-bit register simultaneously. For example: 16 elements in a register A may be added together with 16 elements in a register B in a single instruction as shown in Figure 2.6.

Several versions of SSE have been released(SSE, SSE2, SSE3, SSSE3 and SSE4) and together they cover support for a wide range of arithmetic operations and logical comparisons [16]. The instruction set also contains an instruction for performing SAD on two registers of 16 unsigned 8-bit values. However, there are no instructions for multiplying the elements in two registers of 16 unsigned 8-bit values, thus making SSE less suitable for optimizing the sum of squared differences and normalized cross-correlation. The applications in this thesis only use SSE instructions from SSE and SSE2.

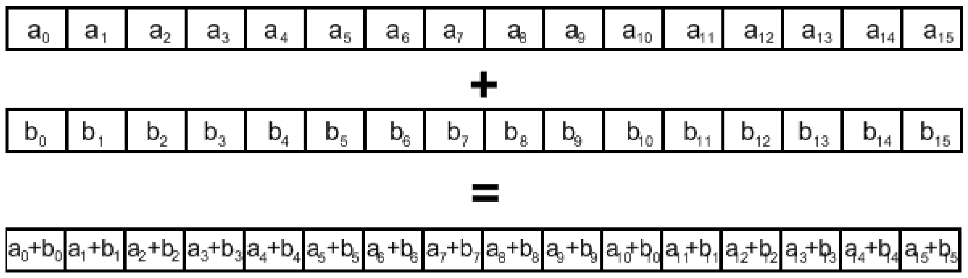

a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12} a_{13} a_{14} a_{15}
+
b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7 b_8 b_9 b_{10} b_{11} b_{12} b_{13} b_{14} b_{15}
=
a_0+b_0 a_1+b_1 a_2+b_2 a_3+b_3 a_4+b_4 a_5+b_5 a_6+b_6 a_7+b_7 a_8+b_8 a_9+b_9 $a_{10}+b_{10}$ $a_{11}+b_{11}$ $a_{12}+b_{12}$ $a_{13}+b_{13}$ $a_{14}+b_{14}$ $a_{15}+b_{15}$

Figure 2.6: SSE instructions may execute up to 16 additions in a single instruction.

Chapter 3

Methods

3.1 Real-time Contour Tracking Library

Real-time Contour Tracking Library (RCTL) is a software library for adapting deformable structures in volumetric data. The library was developed by Fredrik Orderud as a part of his PhD. at NTNU, and have been described in several papers [20, 21, 22]. RCTL provides a fully automated tracking system, and it uses a subdivision model combined with a Kalman-filter based tracking framework.

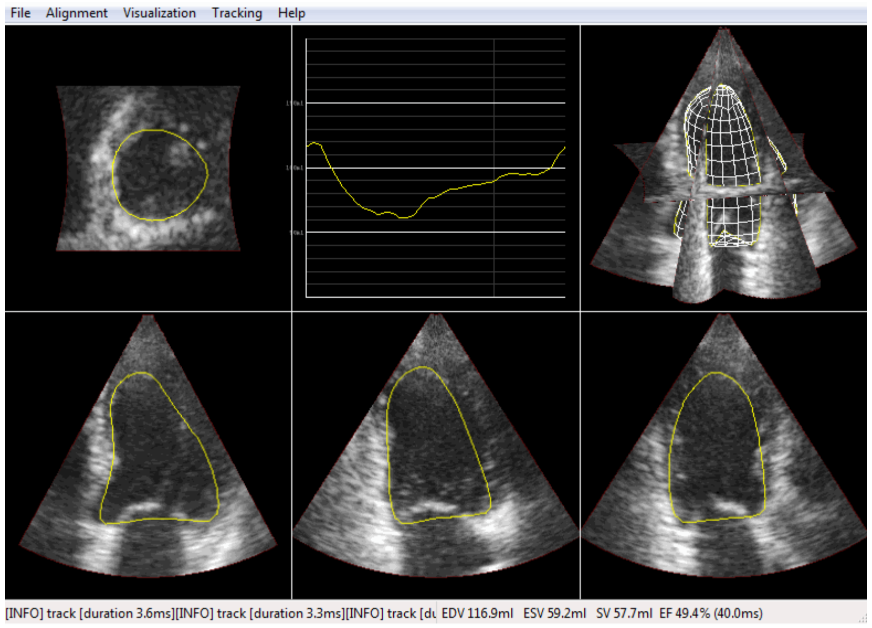
RCTL was initially developed for tracking the left ventricle in images from 3D echocardiogrphy. However, it has later been extended to be able to track other structures, like e.g. the right ventricle [8] and the bladder.

The core functionality of RCTL is written in C++, and the framework has a graphical interface for both 2D and 3D tracking. In addition it is equipped with a MEX-based front end for MATLAB intended for batch-mode tracking.

RCTL performs tracking directly in the image data from ultrasound recordings. The data are represented as grayscale images with 256 intensity levels, and thereby require 8 bits for each voxel.

Because RCTL is designed to perform tracking in real-time, computational efficiency is considered an important aspect.

In this thesis an SSE optimized implementation of block-matching is integrated in the speckle tracking module of RCTL, and RCTL is the system



```
[INFO] track [duration 3.6ms][INFO] track [duration 3.3ms][INFO] track [du  EDV 116.9ml   ESV 59.2ml   SV 57.7ml  EF 49.4% (40.0ms)
```

Figure 3.1: Screenshot from the graphical front-end of the Real-time Contour Tracking Library. The middle box in the top row show a curve of the measured volume. The upper right box visualizes a 3D model of the tracked structure in addition to four slices from the image data. The remaining four boxes displays each of these image slices.

used for tracking the left ventricle in the simulated 3D ultrasound image data.

RCTL uses a Kalman tracking framework. The framework is used together with a deformable model. This deformable model can either be made up by flat polygon surfaces or by Doo-Sabin subdivision surfaces. In either case the surface consists of several control vertices and each vertex is allowed to move in any direction and thereby alter the shape and size of the surface. The points in this model are used as a base for the Kalman tracking [20].

3.1.1 Kalman Filter

The tracking in the Kalman framework can be divided into 5 steps as shown in Figure 3.2. These steps are performed for each frame in the tracking [21].

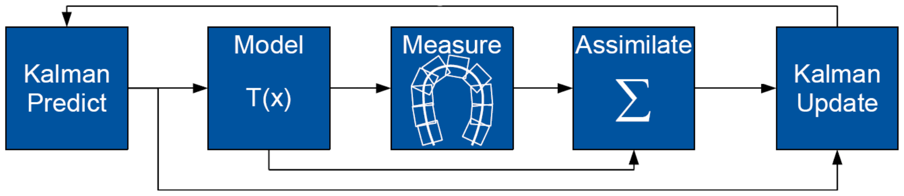


Figure 3.2: Overview of the steps that are performed for each frame in the Kalman filter tracking framework. Figure adapted from [21].

- *State Prediction:* The first step is state prediction. A simple kinematic model is used to predict the state step [22].
- *Evaluation of the deformable model:* The predicted state from the Kalman filter is evaluated in regard to the deformable surface. Local surface points are calculated and fitted to a global model [21].
- *Image measurements:* Edge detection and/or speckle tracking are used to update the displacement of the control points.
- *Measurement assimilations:* Outlier rejection is performed and measured results are assimilated.
- *Measurement update:* Updated state estimate is computed, based on prediction and measurements. Creation of updated surface model.

3.1.2 Image measurements

Image derived measures are used to estimate 3D displacement vectors for a set of predefined surface points. These computations are performed by either edge detection, speckle tracking or a combination of the two.

Speckle tracking is performed around each of the predicted surface points. This process is divided into two steps. First 3D block matching is used with a sum of absolute differences (SAD) matching metric to estimate the best integer voxel displacement. Then Lucas-Kanade optical flow estimation is used to find the best match at a sub-sample level. The search window in the block matching is specified with a fixed size in millimeters in the tracking configurations. The kernel block can be specified to have a fixed size in either millimeters or voxels.

To make the block-matching run efficiently, the SAD was implemented using SIMD instructions. However, these optimizations had only been implemented for kernel blocks with a fixed size of 4x4x4 and 6x6x6 voxels.

3.2 Implementation details

In this thesis optimizations for block matching were integrated in the Realtime Contour Tracking Library(Section 3.1). The implementation used is based on the work performed during the Specialization Project in Complex Computer Systems, fall 2008 [19]. The optimizations use Streaming SIMD Extensions (SSE) to reduce processing time of computation of the sum of absolute differences (SAD).

3.2.1 SSE optimizations

The optimizations consist of a set of implementations of SAD, optimized for various sizes of the kernel block. All the implementations are written in C++, and instructions from SSE2 are used for the SIMD optimization.

The optimizations are integrated in RCTL in a way that it always will use the most specific implementation.

Memory orientation

The reason why it makes sense to have several different implementations of SAD is because SSE (Section 2.5.1) require data simultaneously processed to be adjacent in the memory and properly aligned. Different block sizes make different ways of traversing the data more effective.

While standard instructions are only able to perform SAD on one pair of 8-bit elements in one instruction, SSE instructions are able to process 16 pairs of 8-bit elements in a single instruction. This gives a theoretic possibility of a 16 times increase in processing speed. However, accessing and reading the correct elements also require time. With standard instructions only two 8-bit elements need to be read for each computation. With SSE instructions 2x16 8-bit elements need to be read. The time needed to read these values is affected by how the data are distributed in the memory. E.g.: Reading 16 consecutive elements is faster than reading two segments of eight elements and reading two segments of eight elements is faster than reading four segments of four elements. For the 128-bit blocks to be written to a 128-bit register all its values have to be written to a memory location with a 16-byte alignment first.

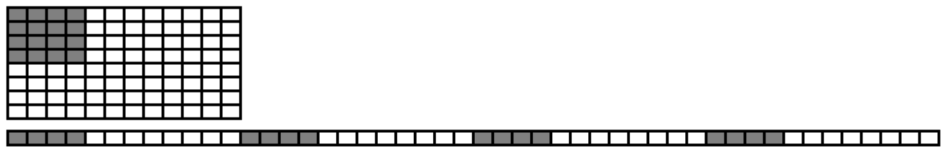


Figure 3.3: Illustration of how a 4x4 block might be distributed in memory.

16x implementation

This implementation only supports kernels with a length along the depth axis of 16 voxels. For each SIMD operation this method will read all the 16 elements from a given row in the kernel block and the 16 corresponding elements in the search window. All the elements for each operation are, in this case, always positioned in the same row, both for the kernel block and search window. This ensures that the 16 elements are stored in a single block, and thus are possible to fetch with a single lookup in the memory. Figure 3.4 illustrates how the block is accessed, first row by row then plane by plane. This method have shown the greatest increase in speed.

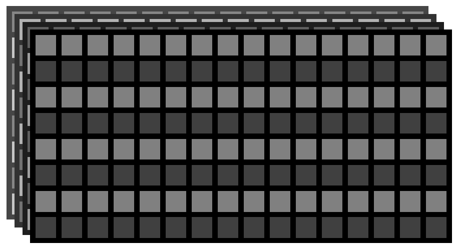


Figure 3.4: The 16x implementation reads one full row of 16 elements at a time.

Modulo 4 implementation

This implementation supports blocks with lengths along the depth and azimuth axis divisible by four. There are no restrictions for the length of the elevation axis. In this method each plane is considered as a tile-set of squares consisting of 4x4 elements, where the elements in each square is processed simultaneously as illustrated in Figure 3.5. This method requires four read operations to access each group of 16 elements from the kernel block, and four operations for reading 16 elements from the search window.

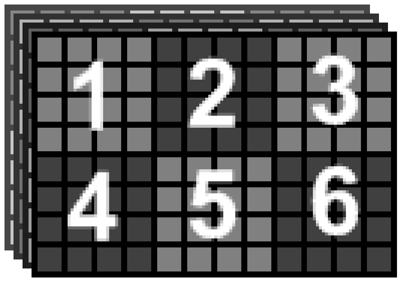


Figure 3.5: The modulo 4 implementation reads segments in each plane as blocks of 4x4 elements.

General implementation

This implementation is designed to handle all possible three-dimensional kernel sizes. It works similar to the modulo 4 implementation in the way that it traverse each plane as a tile-set of 4x4 squares. The process of computing a SAD value in the general implementation is divided in three steps:

1. As much as possible of the plane is considered partitioned in squares of 4x4 elements. This corresponds to Area A in Figure 3.6. This part is processed in the same way as in the modulo 4 implementation.
2. The end of the columns (Area B in Figure 3.6) are processed similar to Area A. The difference here is that extra rows of zeros are added to form squares of 4x4 elements. These extra rows have to always be matched against zeroes instead of values from the search window. An exception from this row padding is when Area B only consists of one

row. Then all the elements are processed as single elements without SIMD optimization.

3. The end of the rows are processed as single elements without SIMD optimization. (Area C in Figure 3.6.)

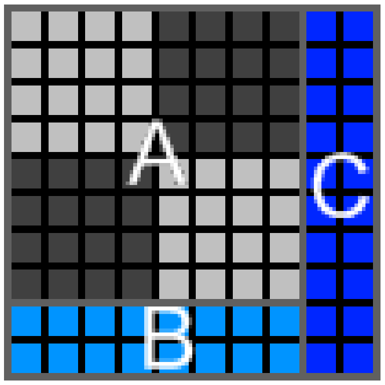


Figure 3.6: The general implementation process each plane of the kernel as three different sections. A: a tile-set of squares of 4x4 elements. B: the end of the columns. C: the end of the rows.

Chapter 4

Test setup

To measure the accuracy of the tracking, simulated ultrasound recordings were used. These recordings are computer generated image data that resemble the result of a 3D ultrasound recording of one cardiac cycle of the heart. The advantage of using simulated recordings is the availability of the ground truth displacement values for each speckle. For clinical recordings there are no such reference.

Tracking were performed on each recording with various parameters for the block-matching.

All the recordings have been tracked in the RCTL framework with an SSE optimized computation of SAD. The tests have been written as scripts in MATLAB and have been run through the MATLAB front-end of RCTL. A laptop with an Intel Core2 Duo T7500 processor at 2.2GHz and 4GB RAM have been used for all the tests.

4.1 Data description

Three datasets were used in the tests. Each resembles the result of a 3D ultrasound recording of one cardiac cycle of the left ventricle.

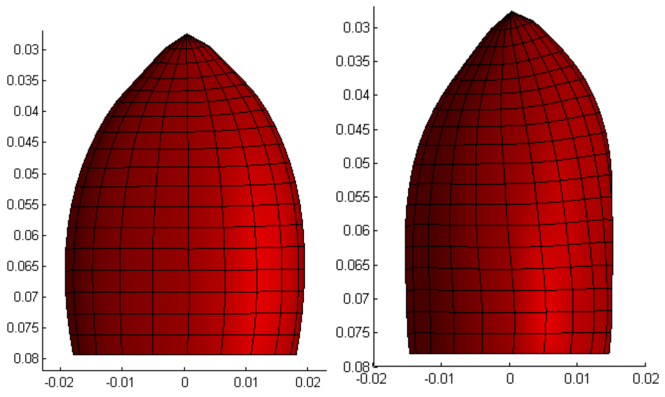
Two of them are generated with the FUSK (Fast Ultrasound Simulation using K-space) generator [13] and the last one is generated with the COLE method. Both methods are based on convolving a set of point scatterers with a point spread function. FUSK operates in the frequency domain while COLE uses the spatio-temporal domain [10]. The positions of the point scatterers are based on finite element simulations the left ventricle.

	Frames	Ranges	Beams	Planes
FUSK data set 1	21	343	149	117
FUSK data set 2	21	343	157	123
COLE data set	21	361	157	123

Table 4.1: Specifications of the dimensions for the different data sets used in the testing.

FUSK data set 1

The first FUSK data set uses a model with an ellipsoidal shape of the myocardium. An antero-apical infarction is simulated in the data set. The image data contain no additional artifacts.



(a) End-diastolic view (b) End-systolic view

Figure 4.1: Model used for the ground truth for first FUSK data set.

FUSK data set 2

The second FUSK data set uses a model with an ellipsoidal shape of the myocardium with a simulated antero-apical infarction like the first data set. However, this data set also includes features of the right ventricle and contains additional noise.

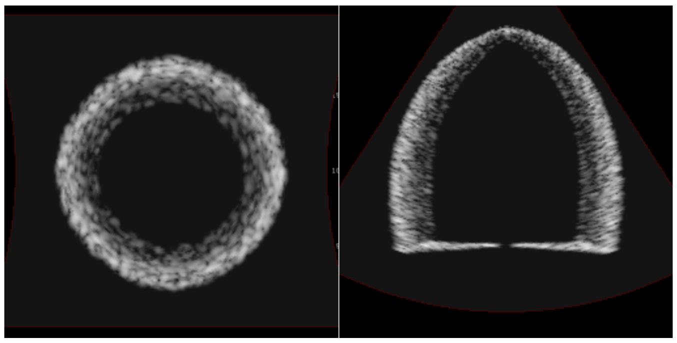
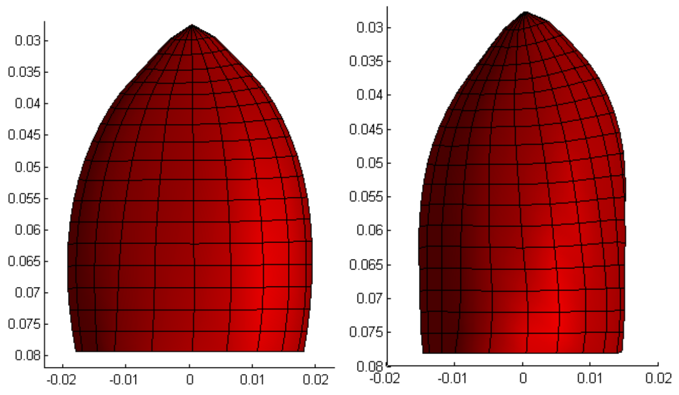


Figure 4.2: Intersection slices of the ultrasound data for the first FUSK data set.



(a) End-diastolic view (b) End-systolic view

Figure 4.3: Model used for the ground truth for the second FUSK data set and the COLE data set.

COLE data set

The COLE data set uses the same ground truth as FUSK data set 2, except that the elevation axis is inverted, resulting in a mirrored model. This data set also contain features of a right ventricle, but no extra noise is added.

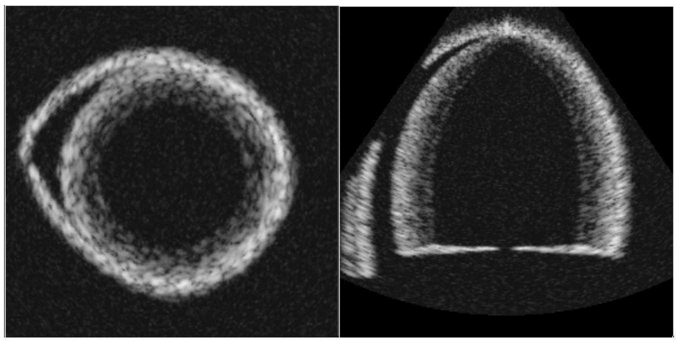


Figure 4.4: Intersection slices of the ultrasound image from the second FUSK data set.

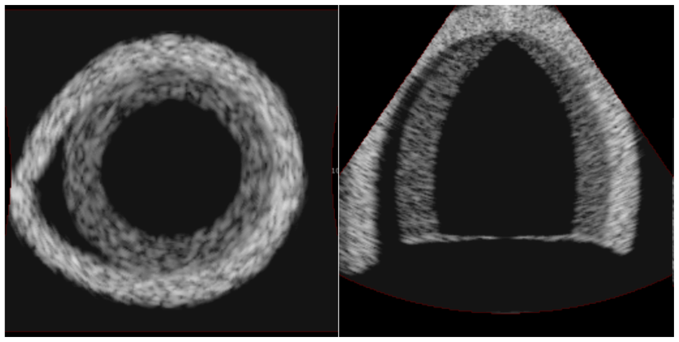


Figure 4.5: Intersection slices of the ultrasound image from the COLE data set.

4.2 Tracking specifications

A combination of edge detection and speckle tracking have been used for the image measurements. This combination has been shown to more accurately track myocardium movement, as compared to employing only speckle tracking measures [20]. The latter case is more susceptible to drift artifacts. The configuration file used for the tracking can be seen in Appendix A.

Variations in block sizes

The data sets have been tracked with various block sizes specified in beamspace coordinates, which means the blocks have fixed voxel volumes. A total of

63 different block sizes have been tested. With axis lengths in following inclusive intervals:

- *Depth:* [4.. 16] elements
- *Azimuth:* [4.. 8] elements
- *Elevation:* [4.. 8] elements

All the block sizes are listed in Table B.1 in the appendix. A search window of 8mm x 4mm x 4mm have been used in this testing.

Variations in size of search window

The data sets have been tracked with various block sizes combined with four sizes of search windows. The following search windows have been used:

- 6mm x 3mm x 3mm
- 8mm x 4mm x 4mm
- 10mm x 6mm x 6mm
- 12mm x 8mm x 8mm

This specifies the range the block matching is allowed to search in each direction from each control vertex in the deformable model. The block matching is only allowed to search offsets where the entire kernel block is within the range of the search window. The first parameter represents the depth direction in the data set. The second parameter specifies length along the azimuth direction. The third parameter is the length along the elevation axis.

Tracking in various segments of the left ventricle

The accuracy of the tracking have been measured in three different regions of the ventricle:

- Apical area
- Mid-level area
- Basal area

The tracked meshes have been divided into three segments with an equal number of polygons in each.

Tracking with block sizes specified in beamspace and cartesian coordinates

The data sets have been tracked with various block sizes specified in both cartesian and beamspace coordinates. I.e. block sizes with both fixed size in voxels and fixed size in millimeters have been tested. 96 different block sizes with fixed cartesian volumes have been tested. Where the axis lengths are in the following inclusive intervals:

- *Depth:* 1.5 - 4.0 mm
- *Azimuth:* 1.5 - 3.0 mm
- *Elevation:* 1.5 - 3.0 mm

All the cartesian block sizes are listed in Table B.2 in the appendix.

4.3 Error metrics

Two different error metrics have been used in the testing. One for measuring relative error and one for measuring absolute error. The displacement used in the computation of the error metrics is the displacement between the first frame of the recording which is at the end of the diastole to the frame representing the end of the systole.

The error metric for the relative error uses the absolute difference in the length of the displacement vector from the tracking $\mathbf{t}$ and the length of the displacement vector from the ground truth $\mathbf{g}$. This difference is divided by the length of the displacement from the ground truth. This gives the error fraction f for the displacement of each polygon i in the tracked mesh. To avoid that errors from areas with small displacements affect the result too much, a cap have been set at 0.5 for the error fraction. The relative error is computed as an area weighted average of these error fractions.

$$f_i = \frac{|\mathbf{t}_i - \mathbf{g}_i|}{|\mathbf{g}_i|}$$

$$Relative\ error = \frac{\sum f_i A_i}{A_{tot}}$$

The error metric for the absolute error uses the root-mean-square (RMS) of the difference between the displacement of the tracked mesh and the ground truth.

$$RMS\ error = \sqrt{\frac{\sum(|\mathbf{t}_i - \mathbf{g}_i|)^2}{N}}$$

Chapter 5

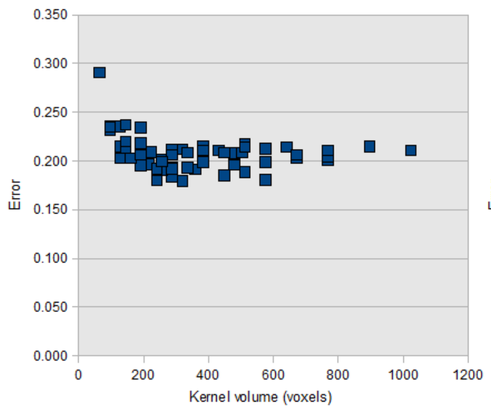
Results and Discussion

5.1 Variations in block sizes

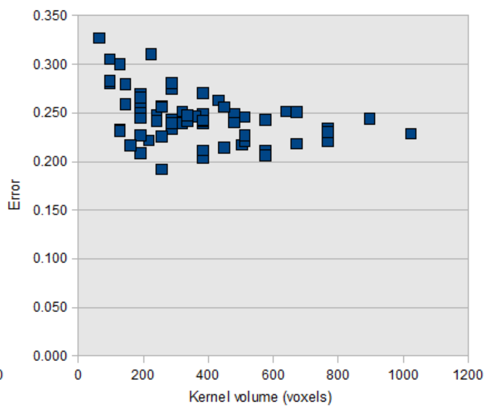
The data sets have been tracked with 63 different block sizes and a search space of 8mm x 4mm x 4mm. All sizes of the kernel blocks are specified in voxels, with lengths of 4 - 16 voxels in the depth direction and 4 - 8 voxels in the other two directions. All the block sizes are listed in Table B.1 in the appendix.

The results for all the data sets show a connection between the volume of the kernel block and the accuracy of the tracking, as shown in Figure 5.1. There was shown little difference in the tracking among the data sets. The most clear difference is that the second FUSK data set (Fig. 5.1b) has a higher variance in the results and has slightly worse accuracy. This seems reasonable as this recording includes additional noise. With the exception of this data set there seems to be little advantage of having kernel blocks with a volume of more than 200 voxels in the current environment.

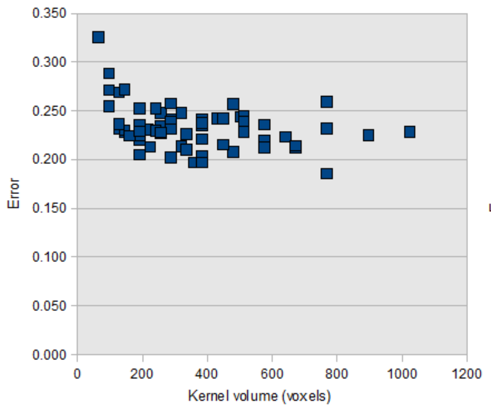
Nearly all block sizes for all data sets with the current search window result in better accuracy than with edge detection alone (Table 5.1). The only exception is the 4x4x4 and 14x4x4 block on the second FUSK data set.



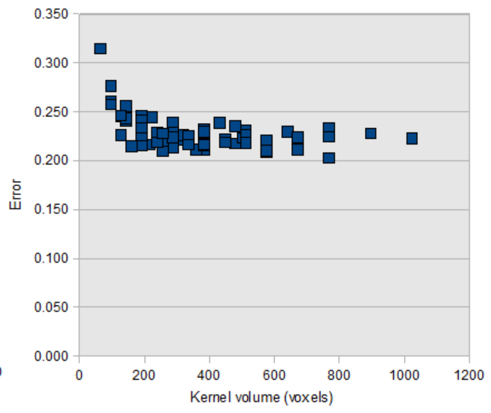
(a) FUSK data set 1



(b) FUSK data set 2



(c) COLE data set



(d) Average

Figure 5.1: Scatter plot measuring the accuracy of block-matching with different sizes of the kernel block. Voxel volume of the kernel block is compared against tracking-error. Tracking-error is area-weighted and measured as the error in end-diastolic to end-systolic displacement between the tracked mesh and the ground truth. The error is normalized to be relative to the displacement in the ground truth. An 8mm x 4mm x 4mm search window is used.

	Relative error	RMS error (mm)
FUSK data set 1	0.4088	2.2614
FUSK data set 2	0.3088	1.4688
COLE data set	0.3336	1.7168
Average	0.3504	1.8157

Table 5.1: Tracking error for the different data sets when using edge detection alone.

5.2 Variations in size of search space

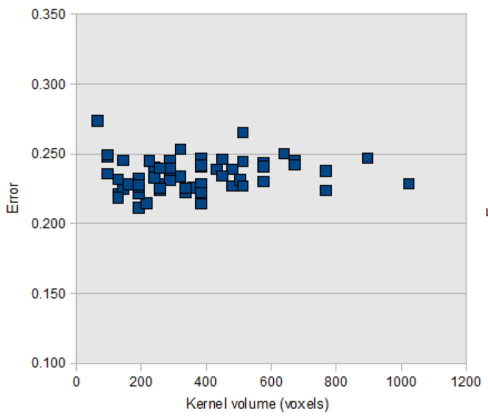
In addition to measure how accurate the tracking is performed with different kernel sizes, variations of the search space have also been tested. Kernel blocks with lengths of 4 - 16 voxels in the depth direction and 4 - 8 voxels in the other two directions have been tested in combination with following sizes of search windows:

- 6mm x 3mm x 3mm
- 8mm x 4mm x 4mm
- 10mm x 6mm x 6mm
- 12mm x 8mm x 8mm

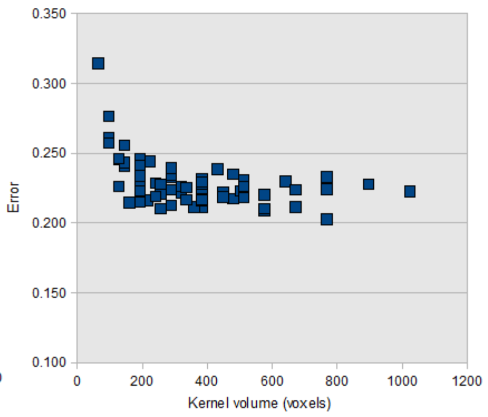
The first parameter is length along the depth axis and the two other parameters are length along the azimuth and elevation axis.

In Figure 5.2 one can see that the kernel block size has bigger impact on the tracking accuracy when the search window is larger. With the smallest search area (Fig. 5.2a), the size of the kernel seems to have no effect at all within the scope of the test cases, but it still performs a lot better than with edge detection alone.

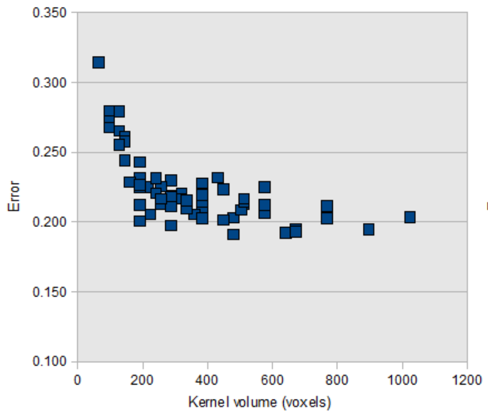
There is also a trend that larger search areas seem to cause worse tracking with small kernel blocks. Since the displacement of any control point between two consecutive frames never exceed 2mm along any of the axes, there is reason to believe that a larger search space will make the appearance of false best matches more likely. However, the results from the tracking with the larger search windows surpass the results from the smaller windows.



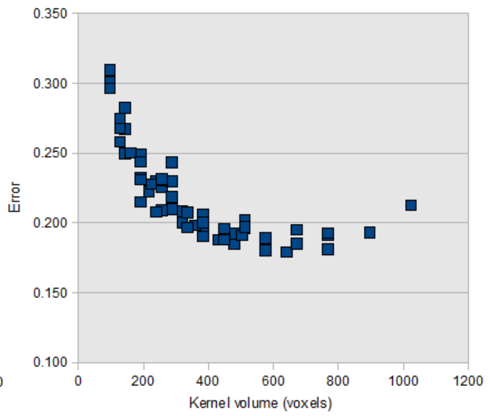
(a) 6mm x 3 mm x 3 mm



(b) 8mm x 4 mm x 4 mm



(c) 10mm x 6 mm x 6 mm



(d) 12mm x 8 mm x 8 mm

Figure 5.2: Scatter plot measuring the accuracy of block-matching with various size of the kernel block compared with different sizes of search area. Voxel volume of the kernel block is compared against tracking-error. Tracking-error is area-weighted and measured as the error in end-diastolic to end-systolic displacement between the tracked mesh and the ground truth. The error is normalized to be relative to the displacement in the ground truth.

5.3 Tracking-differences in various segments of the left ventricle

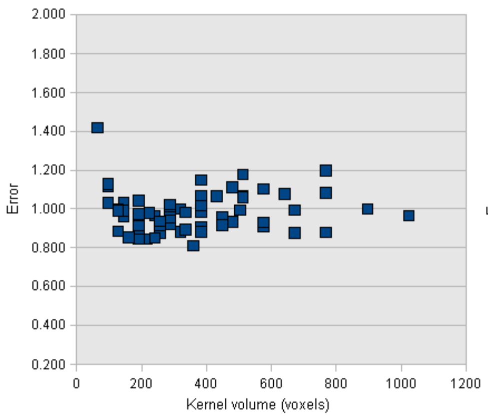
The results from the tracking have also been analyzed with respect to how accurate they are in different regions of the ventricular model. The tracked meshes have been divided in three equal segments with respect to the number of polygons. They are representing the apical area, the mid-level area and the basal area of the ventricle.

In Figure 5.3 one can see that the tracking performs worst in the apical area even though the apical area are exposed for least movement. The mid area shows only slightly better accuracy. Both the mid and apical area show a large spread in the results among the large block sizes. The tracking also becomes slightly worse for the largest cases. The basal area is the only segment were we see a clear improvement in tracking-accuracy for larger blocks, even though this area also yields the best accuracy for most of the smaller cases.

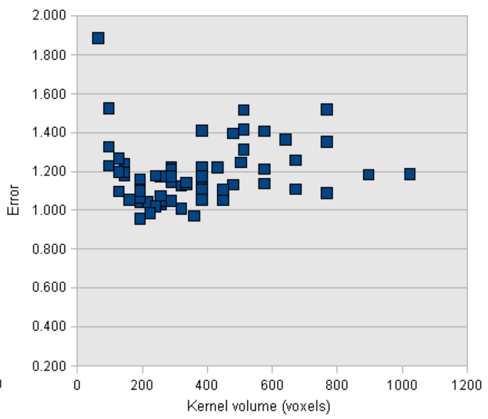
5.4 Comparison between accuracy and run-time

Because of how the SSE-optimizations are implemented there are more aspects of the size of the kernel blocks that just the total volume that affects the processing time of the block-matching.

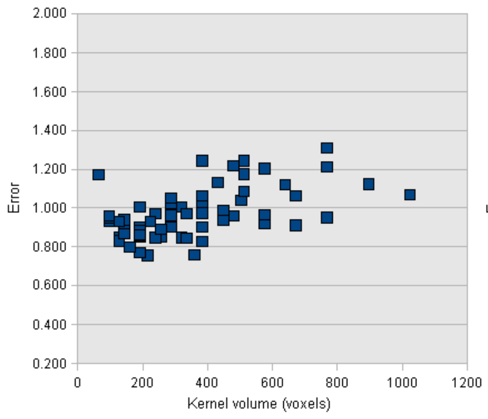
When comparing the run-time, in the tests, with the kernel volume there are some block sizes that distinguish themselves from the other as shown in Figure 5.4. Block sizes with a length of 16 voxels along the depth axis are easy to optimize with SSE and are optimized effectively in the implementations used. Figure 5.5 shows that the tracking performed with the block sizes with a depth of 16 voxels have pretty good ratio between accuracy and run-time, but the results are not exceptional. Several other test cases show comparable results.



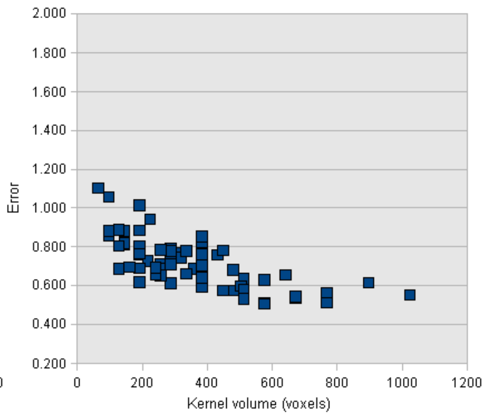
(a) Total area



(b) Apical area



(c) Mid area



(d) Basal area

Figure 5.3: Scatter plot measuring the accuracy of block-matching with various size of the kernel block in different regions of the model. Voxel volume of the kernel block is compared against tracking-error. Tracking-error is measured as the root-mean-square (RMS) of the difference (in millimeters) in end-diastolic to end-systolic displacement between the tracked mesh and the ground truth. An 8mm x 4mm x 4mm search window is used.

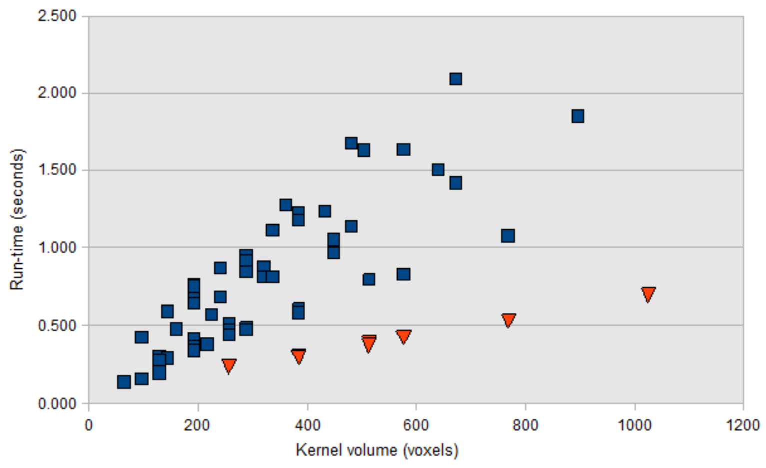


Figure 5.4: Scatter plot comparing the run-time of the speckle tracking with the volume of the kernel block in the block matching algorithm. Different sizes of kernel blocks are used. Results from kernel blocks with a length of 16 voxels along the depth axis have been marked as triangles. An 8mm x 4mm x 4mm search window is used. Run-time is measured in time per frame of the tracking.

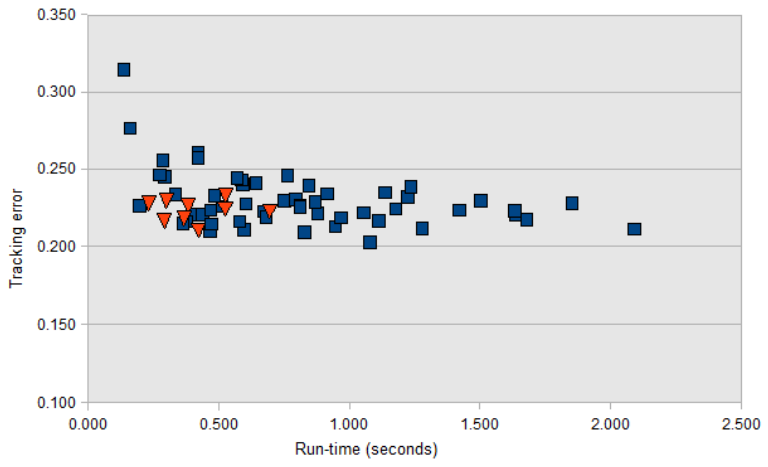


Figure 5.5: Scatter plot comparing the run-time of the speckle tracking with the accuracy of the tracking. Different sizes of kernel blocks are used. Kernel blocks with a length of 16 voxels along the depth axis have been marked as triangles. An 8mm x 4mm x 4mm search window is used. Run-time is measured in time per frame of the tracking.

5.5 Differences between block sizes specified in beamspace and cartesian coordinates

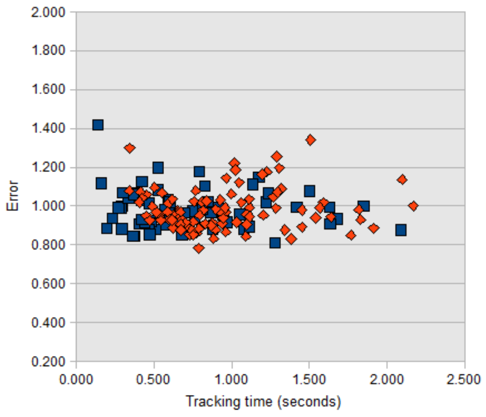
The data sets have been tracked with 63 block sizes specified in beamspace coordinates and 96 block sizes specified in cartesian coordinates. The kernel blocks specified in beamspace coordinates have lengths of 4 - 16 voxels in the depth direction and 4 - 8 voxels in the other two directions. The kernel blocks specified in cartesian coordinates have lengths of 1.5 - 4.0 mm in the depth direction and 1.5 - 3.0 mm in the other two directions. All the block sizes used are listed in Table B.1 and Table B.2 in the appendix. A search space of 8mm x 4mm x 4mm have been used.

The results show little difference between the accuracy of the tracking with kernels specified in beamspace and cartesian coordinates (Figure 5.6). The apical and mid-level regions show practically no improvement in accuracy when using more computationally costly block-matching parameters. There is also a fairly high variance in the results in these regions. The basal area shows an increase in accuracy for slower block sizes for both the cartesian and beamspace test-cases. Block sizes specified in beamspace coordinates get slightly better tracking accuracy here for the large and medium sized blocks, but the cartesian test-case produced clearly worse results for the small block sizes. This last difference was expected as the block sizes specified in cartesian coordinates have a very low resolution along the elevation and azimuth axes in the basal area.

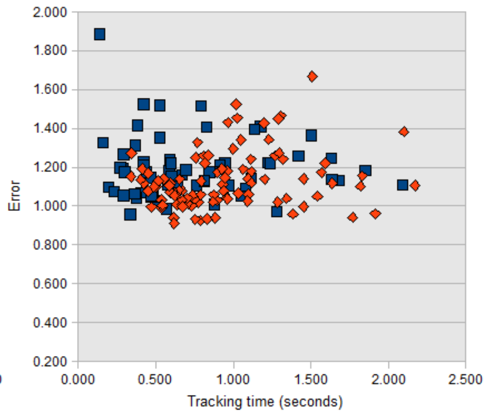
5.6 Sources of error

One weakness with the results presented in this thesis is that they are based on a very low number of data sets. With only three data sets, there is a chance that the result shown just reflects the properties of with a given test set and not echocardiographic images of the left ventricle in general.

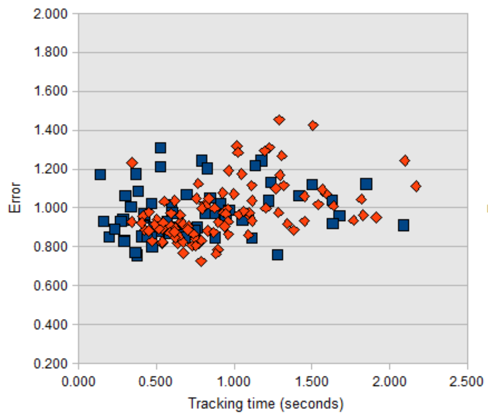
An other unfavorable feature is that the ventricles in the data sets are very similar in shape. Two of the data sets are based on the same ground truth while the third have small differences in the contraction of the model. The shape of the ventricles in the data sets is also less complex than the shape of real ventricles.



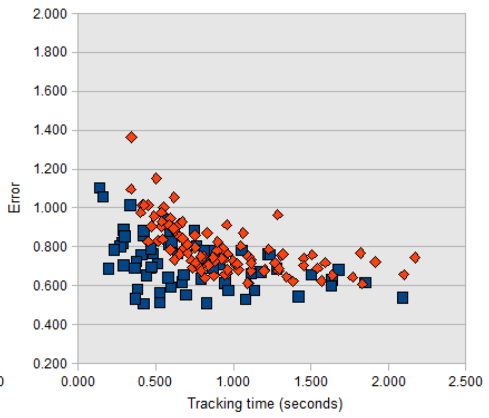
(a) Total area



(b) Apical area



(c) Mid area



(d) Basal area

Figure 5.6: Comparison between accuracy and run-time of block-matching with sizes of kernel blocks specified in beamspace coordinates and cartesian coordinates. Beamspace coordinates are marked in blue and cartesian space coordinates are marked in red. Tracking error is measured as the root-mean-square (RMS) of the difference (in millimeters) in end-diastolic to end-systolic displacement between the tracked mesh and the ground truth. An 8mm x 4mm x 4mm search window is used. Run-time is measured in time per frame of the tracking.

Chapter 6

Conclusion

Optimizations for block matching in 3D speckle tracking have been integrated into the Real-Time Contour Tracking Library (RCTL). These optimizations enable the use of SSE-instructions for arbitrary block sizes. This results in more possibilities when choosing how the block sizes in the tracking shall be specified.

A study have been conducted on how the specifications of the size of the kernel block in block-matching affect the results of 3D speckle tracking of the left ventricle. Tracking results have been measured by both accuracy and computational time.

The results have indicated that block sizes larger than 4x4x4 voxels, which was the initially optimized kernel size for RCTL, are attractive when considering both accuracy and time-efficiency. However, blocks with voxel volumes significantly larger than 200 voxels result in poorer time-accuracy ratio. The benefits achieved by increasing the size of the kernel block were almost exclusively caused by increased accuracy in the basal area of the ventricle.

The tests comparing the kernel blocks specified in cartesian coordinates and the kernel blocks specified in beamspace coordinates showed worse results for cartesian coordinates, when looking at the results for the basal area for the cases with small blocks. Otherwise they showed similar tracking results.

Chapter 7

Further work

Interesting further studies related to the results presented in this thesis are to investigate if the tendencies shown here also applies to real patient data. One will not be able to have the same level of accuracy of the measurements when working with real ultrasound recordings. However, some measurements may be compared with assessments from older 2D tools.

Another interesting aspect is to study how other block-matching related parameters affect the performance. E.g. how to weight the best matches from the block-matching in relation to the background noise.

As the block sizes specified in cartesian coordinates did not perform as well as the ones specified in beamspace coordinates, measures that might improve the processing time for these cases will be interesting to investigate.

One problem with the cartesian blocks is that, for most of the cases tested in thesis, the resolution of the kernel blocks in the basal area is very low along the azimuth and elevation axes. Most of the cases tested in this thesis experience block sizes with just 2-3 elements along these axes. One possible solution to this problem that would be interesting to investigate is the effect of introducing a minimum limit of 4 elements for the length of an axis. Block sizes of 4x4 elements along the elevation and azimuth axis have shown satisfying tracking results in the basal area and are well optimized with respect to processing time.

The SSE optimized block-matching is very efficient for kernel blocks with a length of 16 elements along the depth axis. This efficiency should also be possible to exploit to a certain degree for larger blocks, by using the same principle as the ”16x-optimization” for the first 16 elements of each row in the kernel block. This should improve the processing time for blocks specified in cartesian coordinates, as many of the cases presented in this thesis have more than 16 elements along the depth axis.

Bibliography

[1] M. E. Anderson and G. E. Trahey. A seminar on k-space applied to medical ultrasound. Department of Biomedical Engineering, Duke University, 2006.

[2] B. Angelsen and H. Torp. *Ultrasound Imaging - Waves, Signals and Signal processing in Medical Ultrasonics, Vol. I and Vol. II.* Emantec, www.ultrasoundbook.com, 2000.

[3] J. C. Bamber and C. Daft. Adaptive filtering for reduction of speckle in ultrasonic pulse-echo images. *Ultrasonics*, 24:41–44, Jan. 1986.

[4] M. D. Cerqueira, N. J. Weissman, V. Dilsizian, A. K. Jacobs, S. Kaul, W. K. Laskey, D. J. Pennell, J. A. Rumberger, T. Ryan, and M. S. Verani. Standardized myocardial segmentation and nomenclature for tomographic imaging of the heart: a statement for healthcare professionals from the cardiac imaging committee of the council on clinical cardiology of the american heart association. *Circulation*, 105(4):539–542, Jan. 2002.

[5] X. Chen, H. Xie, R. Erkamp, K. Kim, C. Jia, J. M. Rubin, and M. O'Donnell. 3-d correlation-based speckle tracking. *Ultrasonic Imaging*, 20:151–159, 2005.

[6] J. Crosby, B. H. Amundsen, T. Hergum, E. W. Remme, S. Langeland, and H. Torp. 3D speckle tracking for assessment of regional left ventricular function. *Ultrasound in Medicine and Biology*, 35(3):458–471, 2008.

[7] R. Duncan. A survey of parallel computer architectures. *IEEE Computer*, 23(2):5–16, Feb. 1990.

[8] A. B. Engås. Segmentation of right ventricle in 3D ultrasound recordings. Master's thesis, NTNU, 2008.

[9] M. J. Flynn. Some computer organizations and their effectiveness. *IEEE Transactions on Computers*, 21(9):948–960, 1972.

[10] H. Gao, T. Hergum, H. Torp, and J. D'hooge. Comparison of the performance of different tools for fast simulation of ultrasound data. In *IEEE Ultrasonics symposium*, 2008.

[11] H. Gray. *Anatomy of the Human Body.* Lea & Febiger, bartleby.com, 20th edition.

[12] T. Helle-Valle, J. Crosby, T. Edvardsen, E. Lyseggen, B. H. Amundsen, H.-J. Smith, B. D. Rosen, J. A. Lima, H. Torp, H. Ihlen, and O. A. Smiseth. New noninvasive method for assessment of left ventricular rotation: Speckle tracking echocardiography. *Circulation*, 112:3149–3156, Nov. 2005.

[13] T. Hergum, J. Crosby, M. Langhammer, and H. Torp. The effect of including fiber orientation in simulated 3d ultrasound images of the heart. In *IEEE Ultrasonics Symposium*, pages 1991–1994, 2006.

[14] S. Holm. Medisinsk ultralydavbildning. *Fra fysikkens verden*, (1), 1999.

[15] IntelCorporation. Block-matching in motion estimation algorithms using Streaming SIMD Extensions 3. Intel Software Network, 2003.

[16] IntelCorporation. Intel C++ compiler, user and reference guides. http://www.intel.com/software/products/compilers/docs/clin/main_cls/, 2008. [Online; accessed 10.06.2009].

[17] K. Kaluzynski, X. Chen, S. Emelianov, A. Skovoroda, and M. O'Donnell. Strain rate imaging using two-dimensional speckle tracking. *Ultrasonics, Ferroelectrics and Frequency Control, IEEE Transactions on*, 48(4):1111–1123, July 2001.

[18] J. Meunier. Tissue motion assessment from 3D echographic speckle tracking. *Physics in Medicine and Biology*, 43(5):1241–1254, 1998.

[19] K. E. Nielsen. Performance optimization of 3D speckle tracking in echocardiography. Specialization Project in Complex Computer Systems, NTNU, 2008.

[20] F. Orderud, G. Kiss, S. Langeland, E. W. Remme, H. Torp, and S. I. Rabben. Combining edge detection with speckle-tracking for cardiac

strain assessment in 3D echocardiography. In *IEEE Ultrasonics symposium*, 2008.

[21] F. Orderud, G. Kiss, S. Langeland, E. W. Remme, H. Torp, and S. I. Rabben. Real-time left ventricular speckle-tracking in 3d echocardiography with deformable subdivision surfaces. In *MICCAI Workshop on Analysis of Medical Images*, 2008.

[22] F. Orderud and S. I. Rabben. Real-time 3d segmentation of the left ventricle using deformable subdivision surfaces. In *IEEE Conference on Computer vision and pattern recognition*, 2008.

[23] I. E. G. Richardson. *H.264 and MPEG-4 video compression: Video coding for next-generation multimedia*, pages 225–230. Chichester: John Wiley & Sons Ltd., 2003.

[24] R. R. Seeley, T. D. Stephens, and P. Tate. *Essentials of anatomy & physiology*. McGraw-Hill, 6th edition, 2007.

[25] Y. Shi and H. Sun. *Image and Video Compression for Multimedia Engineering: Fundamentals, Algorithms, and Standards*, pages 221–248. CRC Press, 2000.

[26] Statistisk Sentralbyrå. Dødsårsaker, 2007. http://www.ssb.no/dodsarsak/, Apr. 2009. [Online; accessed 10.06.2009].

[27] Wikipedia. Diagram of the human heart. http://en.wikipedia.org/wiki/File:Diagram_of_the_human_heart_(cropped).svg, 2006. [Online; accessed 10.06.2009].

Appendix

Appendix A

Tracking configuration

```
<!-- Realtime Contour Tracking Library - RCTL -->
<!-- Configuration file for 3D LV tracking     -->
<rctl>

<filter type="ekf">

<transform type="full" state="0 0.028 0  0.050" regularization="
    0.005  0.015  0.005" noise="0.15 0.40 3.13">
  <model type="cell" parameters="3d/LV_Strain.h5m" elastic="true
      " state="" stiffness="0" resolution="4" regularization="
      0.01" noise="1.0">

    <edge type="dual" parameters="0.001 30 50" samples="20"
        spacing="0.0005" noise="0.02" threshold="5" outlier="0.01
        "/>
    <edge type="peak" parameters="-0.001 30 50" samples="20"
        spacing="0.0005" noise="0.02" threshold="5" outlier="0.01
        "/>

    <track type="raw sad" parameters="0.001 4"
    search="XX XX XX" kernel="XX XX XX" spacing="" noise="0.02"
        threshold="0.55" outlier="0.003"/>

  </model>
</transform>

</filter>

</rctl>
```

Listing A.1: The configuration file used in the tracking.

The fields in the configurations file are marked with XX are fields that have different values in different test cases. The search field specifies the distance (in meters) the block matching algorithm is allowed to search in each direction along the depth azimuth and elevation axes. The kernel field specifies the size of the kernel block. These parameters may be given in either meters or voxels.

Appendix B

Block sizes

4x4x4	8x4x4	12x4x4	16x4x4
4x4x6	8x4x6	12x4x6	16x4x6
4x4x8	8x4x8	12x4x8	16x4x8
4x6x4	8x6x4	12x6x4	16x6x4
4x6x6	8x6x6	12x6x6	16x6x6
4x6x8	8x6x8	12x6x8	16x6x8
4x8x4	8x8x4	12x8x4	16x8x4
4x8x6	8x8x6	12x8x6	16x8x6
4x8x8	8x8x8	12x8x8	16x8x8
6x4x4	10x4x4	14x4x4	
6x4x6	10x4x6	14x4x6	
6x4x8	10x4x8	14x4x8	
6x6x4	10x6x4	14x6x4	
6x6x6	10x6x6	14x6x6	
6x6x8	10x6x8	14x6x8	
6x8x4	10x8x4	14x8x4	
6x8x6	10x8x6	14x8x6	
6x8x8	10x8x8	14x8x8	

Table B.1: List of the different block sizes used. Used in testing of block sizes with fixed voxel volume. The parameters specify the lengths along the depth, azimuth and elevation axes.

1.5 x 1.5 x 1.5	2.5 x 1.5 x 1.5	3.5 x 1.5 x 1.5
1.5 x 1.5 x 2.0	2.5 x 1.5 x 2.0	3.5 x 1.5 x 2.0
1.5 x 1.5 x 2.5	2.5 x 1.5 x 2.5	3.5 x 1.5 x 2.5
1.5 x 1.5 x 3.0	2.5 x 1.5 x 3.0	3.5 x 1.5 x 3.0
1.5 x 2.0 x 1.5	2.5 x 2.0 x 1.5	3.5 x 2.0 x 1.5
1.5 x 2.0 x 2.0	2.5 x 2.0 x 2.0	3.5 x 2.0 x 2.0
1.5 x 2.0 x 2.5	2.5 x 2.0 x 2.5	3.5 x 2.0 x 2.5
1.5 x 2.0 x 3.0	2.5 x 2.0 x 3.0	3.5 x 2.0 x 3.0
1.5 x 2.5 x 1.5	2.5 x 2.5 x 1.5	3.5 x 2.5 x 1.5
1.5 x 2.5 x 2.0	2.5 x 2.5 x 2.0	3.5 x 2.5 x 2.0
1.5 x 2.5 x 2.5	2.5 x 2.5 x 2.5	3.5 x 2.5 x 2.5
1.5 x 2.5 x 3.0	2.5 x 2.5 x 3.0	3.5 x 2.5 x 3.0
1.5 x 3.0 x 1.5	2.5 x 3.0 x 1.5	3.5 x 3.0 x 1.5
1.5 x 3.0 x 2.0	2.5 x 3.0 x 2.0	3.5 x 3.0 x 2.0
1.5 x 3.0 x 2.5	2.5 x 3.0 x 2.5	3.5 x 3.0 x 2.5
1.5 x 3.0 x 3.0	2.5 x 3.0 x 3.0	3.5 x 3.0 x 3.0
2.0 x 1.5 x 1.5	3.0 x 1.5 x 1.5	4.0 x 1.5 x 1.5
2.0 x 1.5 x 2.0	3.0 x 1.5 x 2.0	4.0 x 1.5 x 2.0
2.0 x 1.5 x 2.5	3.0 x 1.5 x 2.5	4.0 x 1.5 x 2.5
2.0 x 1.5 x 3.0	3.0 x 1.5 x 3.0	4.0 x 1.5 x 3.0
2.0 x 2.0 x 1.5	3.0 x 2.0 x 1.5	4.0 x 2.0 x 1.5
2.0 x 2.0 x 2.0	3.0 x 2.0 x 2.0	4.0 x 2.0 x 2.0
2.0 x 2.0 x 2.5	3.0 x 2.0 x 2.5	4.0 x 2.0 x 2.5
2.0 x 2.0 x 3.0	3.0 x 2.0 x 3.0	4.0 x 2.0 x 3.0
2.0 x 2.5 x 1.5	3.0 x 2.5 x 1.5	4.0 x 2.5 x 1.5
2.0 x 2.5 x 2.0	3.0 x 2.5 x 2.0	4.0 x 2.5 x 2.0
2.0 x 2.5 x 2.5	3.0 x 2.5 x 2.5	4.0 x 2.5 x 2.5
2.0 x 2.5 x 3.0	3.0 x 2.5 x 3.0	4.0 x 2.5 x 3.0
2.0 x 3.0 x 1.5	3.0 x 3.0 x 1.5	4.0 x 3.0 x 1.5
2.0 x 3.0 x 2.0	3.0 x 3.0 x 2.0	4.0 x 3.0 x 2.0
2.0 x 3.0 x 2.5	3.0 x 3.0 x 2.5	4.0 x 3.0 x 2.5
2.0 x 3.0 x 3.0	3.0 x 3.0 x 3.0	4.0 x 3.0 x 3.0

Table B.2: List of the different block sizes used. Used in testing of block sizes with fixed cartesian volume. All the values are specified in millimeters. The parameters specify the lengths along the depth, azimuth and elevation axes.