Imperial College London

Department of Computing

Motion Stabilisation for Dynamic Medical Image Sequences

by

Fahdi Kanavati

Submitted in partial fulfilment of the requirements for the MSc Degree in Advanced Computing of Imperial College London

September 2013

Abstract

In many medical imaging scenarios dynamic image sequences are acquired from patients during interventions. In these image sequences there is often a significant amount of cardiac and/or respiratory motion and this can make it difficult for the surgeon to navigate and perform intricate procedures. One way to overcome this is through the use of virtual stabilisation. Obtaining good stabilisation largely depends on the motion estimation which is more challenging when it comes to dealing with deformable tissue.

In this project we investigate the post-processing stabilisation technique suggested by Grundmann et al. in [1] for its application on cardiac video sequences. We also explore a possible technique for stabilisation based on manifold learning. An extension of both techniques for predictive stabilisation is also investigated.

The project found that applying 2D stabilisation techniques to deformable surfaces such as that of the heart can lead to a considerable reduction of respiratory and cardiac motion even if the surface deformation is non-linear, however, it is not without a few issues.

Acknowledgement

I would like to express my gratitude to my supervisor, Professor Daniel Rueckert, for the continuous support and invaluable advice that he has provided me throughout the development of this project.

I would like to thank my beloved family for their unconditional support, especially my parents for giving me the opportunity to study at Imperial.

Contents

List of Figures

Chapter 1

Introduction

This chapter introduces the motivation for this research project, its objectives and the work that we have carried out. The chapter ends with an outline of the report structure.

1.1 Motivation

Many surgical procedures are being performed using minimally invasive techniques. Minimally invasive surgery (MIS) or laparoscopic surgery is beneficial for patients as it results in fewer tissue damage, minimal pain and faster recovery time. This type of modern surgery tends to rely more on external imaging devices than it is the case with exploratory surgery. A small camera is typically inserted through a small incision and a video feed is relayed back to the surgeon. Most of the medical image sequences acquired from MIS tend to have a significant amount of cardiac and/or respiratory motion which makes it more difficult and demanding to navigate and perform intricate procedures even for the most skilled surgeon. Applying virtual video stabilisation to the acquired video would be of major benefit to the surgeon.

The aim of video stabilisation is the removal of undesired motion. In computer vision, stabilisation can be achieved by 2D or 3D techniques. There has been a lot of work in regards to digital stabilisation to remove shaking and high frequency jitter introduced by filming with a hand-held camera. The 2D stabilisation algorithm [1] that we mainly investigate in this project was designed to address such types of undesired motion. It is incorporated in the YouTube Video Editor as an enhancement feature to provide users the possibility to remove the camera shake from their videos yielding a more pleasing viewing experience.

Apart from digital video stabilisation there exists two other types of video stabilisation: Mechanical and optical. The former is done by physically moving the actual camera in accordance to the detected motion from sensors (gyroscopes, accelerometers, ...), the latter typically involves a moving lens that adjusts the path of the light before it reaches the digital sensor. Both of these techniques require a specialised set-up.

Video stabilisation can typically achieve really good results when the underlying scene being stabilised contains rigid objects which are necessary for obtaining a good global motion estimate using a linear model. In medical image sequences however, the deformation of the cardiac surface is non-rigid, which makes describing the global motion with a simple linear model a non-trivial task. Nonetheless, considering that the camera is usually close to the filmed surface in the medical imaging scenarios, then the motion, locally if not globally, could be approximated with a linear model. The advantage of using 2D stabilisation techniques is that they offer more chances to be integrated in existing MIS workflows that rely on monoscopic cameras and are also less computationally expensive than alternative 3D techniques.

1.2 Aims

The aims of the project are as follows:

- To investigate the effectiveness of the post-processing 2D stabilisation algorithm [1] at removing cardiac and respiratory motion typically present in cardiac video sequences obtained during MIS.
- To investigate possible alternative means for 2D stabilisation for cardiac sequences.
- To investigate if it is possible to extend the post-processing stabilisation techniques into predictive stabilisation.

1.3 Contributions

- An implementation of the L1 optimal paths stabilisation algorithm [1].
- An implementation of a modified specular highlight detection and in-painting algorithm based on the algorithm in [2].
- An investigation and implementation of a 2D stabilisation technique based on manifold learning.
- A graphical user interface in MATLAB regrouping all the different elements into one location.

1.4 Report Structure

The report is structured as follows:

- Chapter 2 describes the related work.
- Chapter 3 describes some of the relevant background knowledge.
- Chapter 4 describes the theory behind the stabilisation algorithms and specular highlight detection and in-painting.
- Chapter 5 describes the software implementation.
- Chapter 6 contains the first set of experiments and their results.
- Chapter 7 contains the second set of experiments and their results.
- Chapter 8 provides conclusions to the work done in this project and possible future work.
- Appendix provides information for running the code.

Chapter 2

Related Work

This chapter goes through some of the related work in regards to video stabilisation and tracking for deformable tissue.

2.1 Digital Video Stabilisation

Digital video stabilisation is the process of removing undesired motion with image processing techniques. Different techniques have been suggested in the literature, but in essence the process typically consists of the following main steps as identified in [3] and in [1]:

1. Estimating the original camera path from the video using a motion model that describes the overall original motion.

2. Estimating a potentially smooth camera path from the estimated original camera path in the form of motion model.

3. Synthesizing / Re-rending a new video from the smoothed path viewpoint.

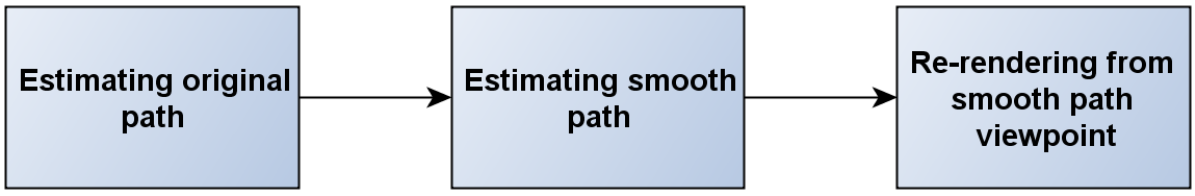


Figure 2.1: Motion stabilisation overview

The end quality of the resulting video depends on the performance of each one of these steps. Motion estimation has to be able to correctly identify the undesired motion that needs to be removed in a scene that could typically consist of different moving objects and clutter.

In current approaches in 2D digital stabilisation, estimation of the camera path is usually done by performing feature tracking between consecutive frames and then estimating the global motion in the scene between each frame in the form of a motion model represented by a geometric transformation, which can consist of translation, scale, rotation, skew and perspective. This type of motion estimation is used in [1] and [3]. The camera path would then be represented as a set of consecutive geometric transformations between consecutive frames. Determining what feature points tracked in the scene that go into estimating the geometric transformation is typically done by applying a statistical algorithm such as the RANSAC algorithm [4] which aims to exclude points that do not fit a given model. It can thus be used to determine tracked points that do not fit the motion model especially those in the scene that belong to moving objects that do not form part of

the rigid background.

For estimating the smooth camera path, Grundmann et al. propose in [1] the use of L1 optimal camera paths which can be obtained by minimising the L1-norm of the first, second and third order derivatives of the desired smooth camera path. They pose the problem as a constrained linear optimisation program. They favour minimising the L1-norm as opposed to the L2-norm as the latter tends to attempt to satisfy the properties on average (in a least squared sense) which means that the end result will still have small non-zero motion in the direction of the shake, while L1-norm will attempt to satisfy the properties exactly. Alternatively, Matsushita et al. [3] propose to smooth the camera path by temporally smoothing local transformation between neighbouring frames.

Estimating the camera path can also be done using 3D techniques such as Structure from Motion (SfM). 3D stabilisation is then achieved by attempting to simulate what the camera sees from a 3D output path. This method is used by Liu et al. in [5] where they employ a SfM technique to reconstruct the original camera motion, then an idealised camera motion such as a line, low-pass filter or a parabola is fit to the original motion path. They also suggest the use of content preserving warp to fake small viewpoint shifts when re-rendering, however the result is not physically accurate but mostly plausible. A major limitation of this method is that SfM is more computationally expensive than typical 2D video stabilisation.

In most of the methods for video stabilisation, when re-rendering, there is an inevitable decrease in resolution due to the out of bounds regions that are introduced by the motion compensation. As a result the video can be cropped and is either rescaled back up to the original video size or simply re-rendered with the lower resolution. Methods using motion-in-painting are suggested in [3] to overcome the loss of parts of the image but it is subject to undesirable artefacts. Usually, depending on the frame rate and shutter speed of the camera, the stabilised video can exhibit blurring in some frames so in [1] they suggest that a solution would be to introduce blur detection and preserving some of the original motion near the blurred frame at the cost of a smooth path.

2.2 Motion Estimation for Deformable Tissue

Most of the techniques mentioned above are intended with tackling video stabilisation for scenes that have a rigid background which make it easier to get a good motion estimate. The techniques will not usually work as expected when it comes to applying them to medical image sequences that typically consist of non-rigid objects of deformable nature such as that exhibited by the deformation of the cardiac surface. This is mainly due to the more challenging task of obtaining a good model of the motion of the heart, since achieving good stabilisation requires good motion estimation. There exists several techniques that have been proposed to attempt to deal with tracking non-rigid deformable tissue. Before mentioning these techniques we first look at the issues typically encountered in medical imaging scenarios that make tracking difficult.

2.2.1 Issues

The main challenges [6] that are commonly mentioned in existing research for applying vision based techniques directly to medical image sequences are the following:

- The deformable nature of the tissue - the changing visual appearance due to tissue deformation pose a big challenge for feature correspondence between two images. Correspondence is usually an easier task when the features correspond to rigid objects.
- Poor illumination conditions and low contrast of the images.
- Noise introduced by the image acquisition system and image artefacts.

- Specular reflections - the wet nature of the tissue gives rise to view dependent specular reflections, which could result in them being detected as regions of interest by the feature detector.
- Occlusions and dynamic scene changes due to the presence of medical instruments, bleeding and smoke arising from cauterisation.
- The appearance of the tissue can vary from being homogeneous to textured, making it difficult in some regions, especially homogeneous regions, to identify features.

2.2.2 Tracking

Motion estimation requires tracking of the position of landmark features / regions of interest that are easy to identify between consecutive frames. The points can be tracked by either detecting feature points on an image and then tracking where they go in the next image. The other method is to detect features on both of the images then attempting to find the correct correspondence between the points.

Associated to an image feature are what is called a feature descriptor which is a means to identify the characteristics of a given landmark. When dealing with deformable tissue more adapted feature detectors are needed to ensure more robust tracking in the context of MIS. Some feature descriptors can be made transformation invariant (such as scale and rotation) however there are still problems due to the issues mentioned above. Mountney et al. did a study in [7] where they evaluate the performance of different state-of-the-art computer vision feature descriptors for tracking deformable tissue. Yang et al. propose in [8] a probabilistic framework for tracking affine invariant anisotropic regions under different visual appearances during MIS and its performance is compared against that of other region detectors, such as the SIFT (DoG), the Harris-Affine Detector, the Hessian-Affine detector and the MSER (Maximally Stable Extremal Regions). A real-time implementation is achieved using the computational power of graphics processing unit (GPU).

Tracking of deformable tissue can also be done by the use of stereo techniques which can yield a more robust tracking and reconstruction, several approaches are suggested in the literature. Feature based motion tracking of deformable tissue using a combination of two feature detectors is described in [9]. 3D tracking is used in [10] to reconstruct the cardiac surface deformation which is then used to render an augmented reality stabilised view. An algorithm for 3D tracking of the beating heart using a Thin-Plate Splines parametric model is proposed in [11]. Other techniques for tracking the deformation of the heart by Magnetic Resonance imaging tagging have also been proposed [12].

2.3 Conclusion

A limitation of 3D stabilisation techniques is that they are more computationally expensive. Although non-linear models also seem necessary to model accurately the soft-tissue deformation, work such as in [13] have showed that it is possible to locally model the deformation with a linear model. Also previous work [14] that aimed at testing robust real-time feature tracking on the beating heart report that the combination of speeded-up robust features (SURF) and the differential tracking algorithm Lucas-Kanade gave good performance in terms of real-time tracking. In this project we make use of these two finding for the motion estimation in particular for the real-time considerations and the fact that we aim to investigate 2D stabilisation techniques.

In this chapter, we made mention of computer vision techniques related to tracking and stabilisation. We give more details about what they consist of in the subsequent chapter.

Chapter 3

Background

In this chapter we go over some background information that was required for this project.

3.1 Notation

Frame / image is defined as a matrix I of dimensions $H \times W \times C$, where H is the height W is the width and C is the number of channels. A color image in the red, green and blue color mode (RGB) has 3 channels. A grayscale image has 1 channel. A pixel $p(i, j)$ is defined as the entry in the matrix at position $I(i, j, :)$ where : refers to all the channels i.e. if there are 3 channels then the pixel is the resulting combination of the 3 channels. For an 8-bit image, each entry in the matrix can have an intensity value between 0 to 255.

Video is defined as a sequence of equally sized images/frames $I_1, \ldots, I_t, \ldots, I_N$, where I_t denotes the image numbered t and N denotes the total number of frames.

A frame rate is associated to the video which denotes the number of frames per second (fps). It defines the speed at which the sequence of images are to be played.

3.2 Image Features

An image feature can be thought of as being a local part of an image that has meaningful or desirable properties, such as geometric (translation, scale and rotation) and photometric (brightness) invariance. There exists several different detection algorithms each aiming to detect a specific type of features that have a given set of properties. A feature is typically described by its location in the image and by a feature descriptor that describes the characteristics around the detected feature. Local features are usually better since they tend to be robust to occlusions.

Features are commonly used in computer vision for tasks such as motion tracking, object recognition, image alignment, robot navigating, stereo correspondence and 3D reconstruction among other applications.

Types of Image Features

Edges are locations in the image around which the intensity values changes sharply.

Corners capture corner like structures in the patterns of intensities. They do not necessary correspond to the intersecting point of two lines. Corners are typically more stable than edges and therefore are more interesting to track.

Regions of interest / Blobs capture image structures in regions that have a particular set of properties (colour, brightness...) that distinguishes them from the surrounding area. Regions of interest provide more details about regions than edges or corners.

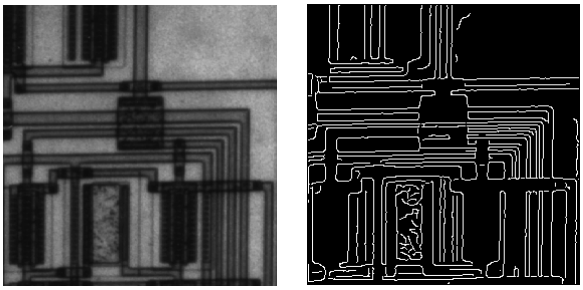


Figure 3.1: Example of edge detection on an image of a circuit. (Source: MATLAB documentation)

3.3 Feature Detection

Feature detection is the method of extracting image features that satisfy a set of criteria that depend on the detection algorithm.

Typical corner detectors rely on a corner response (C) measure of each pixel location in an image. Pixels having a corner value greater than a certain threshold are marked as a corner. Typically such a measure is obtained through the following structure tensor which describes the predominant directions of the gradient in the neighbourhood of a specified point

$$H = \begin{bmatrix} \widehat{I_x^2} & \widehat{I_x I_y} \\ \widehat{I_x I_y} & \widehat{I_y^2} \end{bmatrix} \tag{3.1}$$

H is also called Harris matrix. $\widehat{\cdot}$ denotes the average in the local neighbourhood.

Shi-Tomasi Detector

The Shi-Tomasi Corner Detector [15] is based on the Harris-Stephens Corner Detector [16], the difference is the corner selection criterion. A corner response is assigned to a pixel as $C = min(\lambda_1, \lambda_2)$. λ_1 and λ_2 being the eigenvalues of the Harris matrix.

FAST

Features from Accelerated Segment Test (FAST) [17] is a corner detection method. It relies on local intensity comparison. It has the advantage that it is computationally efficiency. FAST is faster than many other well-known feature detectors. The improved performance is obtained by the use of machine learning making them suitable for real-time performance.

SIFT

Scale Invariant Feature Transform (SIFT) features [18] are obtained by convolution of the image with a difference of Gaussians (DoG) at different scales and key locations having a maxima or a minima are retained. A dominant orientation is assigned at each key location. SIFT descriptors are then made robust to scale and rotation by considering pixels in the local neighbourhood of the key location, by re-sampling local image orientations and blurring.

SURF

SURF (Speeded Up Robust Features) [19] is a feature detector partly inspired from SIFT and is claimed to be faster. The technique relies on integral images for computing the convolutions.

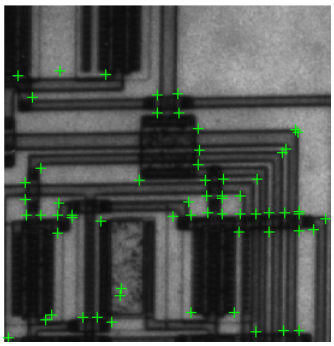


Figure 3.2: Example of detected FAST features (green) on an image of a circuit. (Source: MATLAB documentation)

3.4 Motion Field

The motion field or velocity field is the apparent motion in the scene induced by relative motion between the camera and the scene. It is defined by a set of 2D velocity vectors of each point in an image. Finding the motion field makes it possible to track the motion of a given point across an image sequence.

There exists several different techniques for estimating the motion field and these can be roughly divided into two different types:

Differential techniques are based on the spatio-temporal variation of the intensity of pixels. It is a direct method for motion estimation in the sense that two problems are solved simultaneously: the correspondence between each pair of pixels and the apparent motion.

Matching techniques are an indirect method. First a set of features are selected in two frames. The second step is matching each feature in one frame to the corresponding feature in the other frame. Finding the set of matching correspondences is referred to as the correspondence problem. After matching the features, the motion field is reconstructed.

A motion field can be *dense* i.e. each pixel in an image has a velocity vector or *sparse* i.e. only a set of specific pixels have a motion vector.

Estimating a dense motion field is computationally expensive but it can in some cases be more accurate. The algorithms usually rely on a global smoothness constraint and information for homogeneous parts of an image are filled in from the boundaries. The technique however is more sensitive to noise.

Estimating a sparse motion field is usually less sensitive to noise since it is a local method, it is also more suited for real-time computation. In this project we rely on estimating a sparse motion field.

3.4.1 Optical Flow

Optical flow is a differential technique for estimating the motion field. It is based on the **brightness consistency assumption**, that is, if $I(\mathbf{x}, t) = I(x, y, t)$ represents the intensity located at (x, y) and at time t, then in the ideal case, at the consecutive frame $t + \Delta t$ there should be a displacement $\mathbf{d} = [\Delta x \ \Delta y]^T$ such that

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) \tag{3.2}$$

By making the assumption that the **displacement is small** we can do a Taylor expansion. By ignoring the higher order terms, the above equation becomes equivalent to:

$$\frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt = 0 \tag{3.3}$$

$$\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0 \tag{3.4}$$

$$(\bigtriangledown I)^T \mathbf{v} + I_t = 0 \tag{3.5}$$

where $\mathbf{v} = \begin{pmatrix} \frac{dx}{dt} \\ \frac{dy}{dt} \end{pmatrix}$ and $\bigtriangledown I = \begin{pmatrix} \frac{\partial I}{\partial x} \\ \frac{\partial I}{\partial y} \end{pmatrix}$.

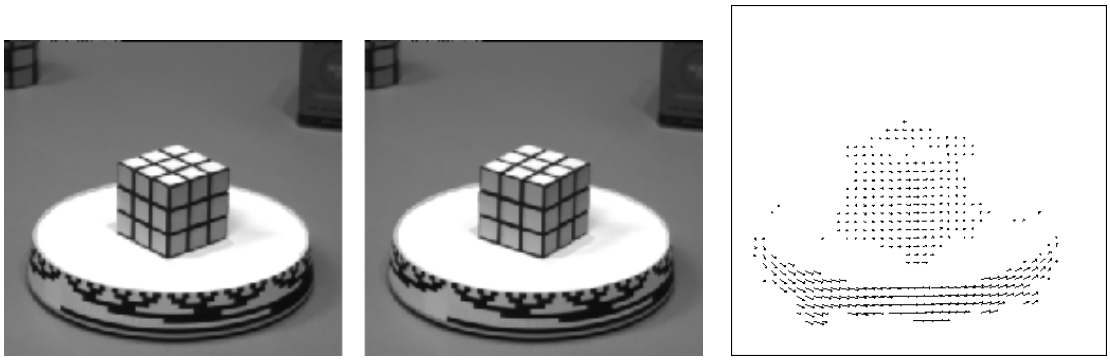
For a pixel p located at (x_0, y_0), the optical flow equation is

$$I_x(p)v_x + I_y(p)v_y = -I_t(p) \tag{3.6}$$

where I_x, I_y and I_t are obtained with the finite difference approximation of the derivatives.

Another assumption for the optical flow is that the surface should be **lambertian** i.e. the apparent brightness does not depend on the viewing angle.

Unfortunately the above assumptions for the optical flow rarely hold. The problem is further complicated due to the presence of noise.



(a) Two images of a rubik's cube taken from slightly different angles. (b) The resulting optical flow.

Figure 3.3: Optical flow. (Source: [20])

Lucas-Kanade Method

The Lucas-Kanade (LK) method [21] for optical flow calculation is based on the assumption that the flow is constant in the **local neighbourhood of a pixel**, by doing so, it is less sensitive to noise. The displacement is calculated based on least squares.

By considering that the displacement between two consecutive image frames is small and approximately constant, for a given small neighbourhood R of size $n = N \times N$ (usually $n = 5 \times 5$) around the pixel located at (x, y), the velocity vector is chosen so as to minimise the sum of squared error e defined by the following:

$$e(\mathbf{v}) = \sum_{p_i \in R} [(\bigtriangledown I(p_i))^T \mathbf{v} + I_t(p_i)]^2 \tag{3.7}$$

In matrix form the optical flow equation we can write 3.6 for pixels $p_1 \dots p_n$ that are located in a neighbourhood R as:

$$A\mathbf{v} = \mathbf{b} \tag{3.8}$$

where

$$A = \begin{pmatrix} I_x(p_1) & I_y(p_1) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{pmatrix}, \mathbf{v} = \begin{pmatrix} v_x \\ v_y \end{pmatrix}, \mathbf{b} = - \begin{pmatrix} I_t(p_1) \\ \vdots \\ I_t(p_n) \end{pmatrix} \tag{3.9}$$

The algebraic solution to 3.8 is given by

$$\mathbf{v} = (A^T A)^{-1} A^T \mathbf{b} \tag{3.10}$$

Usually the assumption that the displacement is small between two frames does not hold as it usually depends on the frame rate of the captured video and the speed at which the point in the image moves. So the distances can be greater than the neighbourhood size. This can be overcome by preforming the LK method on multiple levels of resolution by using image pyramids, where at each level of the pyramid the resolution of the image is reduced by a factor of two compared to the previous one. Applying the LK method starting from the lowest level of resolution then passing the result to the next level to provide an initial estimate of the location of the point.

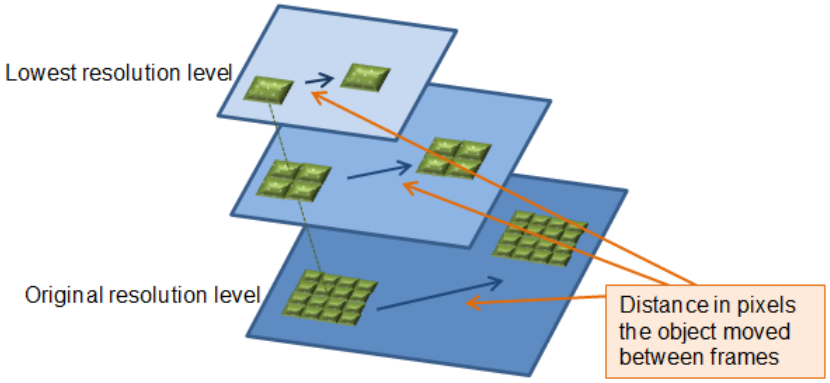


Figure 3.4: Pyramid Levels. (Credit: Matlab documentation on point tracker)

Issues

Issues that could affect the optical flow calculation and that should be dealt with:

Occlusions Objects in the scene can go out of frame or can disappear behind other objects or new elements might appear in the scene.

Noise and Image Distortions The camera sensor can introduce noise which causes changes in the grey values. Also the camera lens can introduce geometrical distortions.

Illumination The grey values could change in the scene due to illumination changes even though nothing moves in the scene. A moving light source in a fixed scene would introduce intensity changes even though there is no motion.

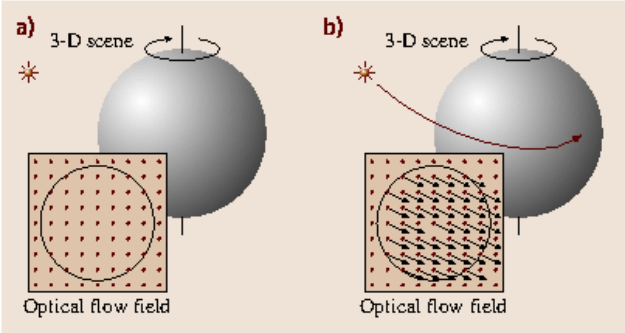


Figure 3.5: In (a) we see that even if the sphere is spinning, there is no resulting optical flow. In (b) we see that a moving light source can introduce apparent motion of the sphere that does not reflect the motion of the sphere.(Source:[22])

3.4.2 Feature Matching

Feature matching techniques are indirect methods for estimating the motion field. It is usually done by detecting a set of features between two frames and attempting to solve the correspondence problem of finding matching pairs of features. The method for feature matching depends on the feature descriptors used. Euclidean distance between the feature descriptors or normalized cross-correlation can be used.

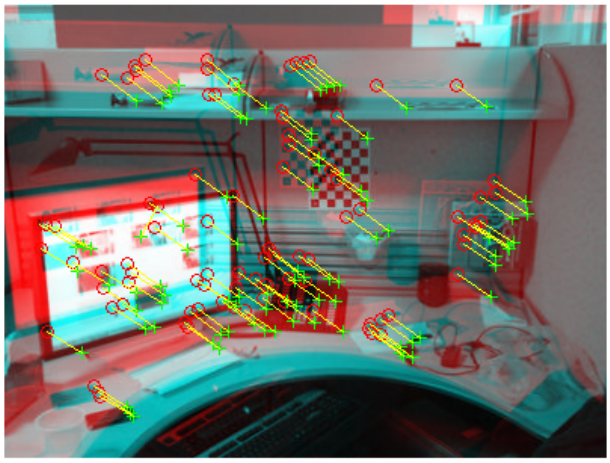


Figure 3.6: Example of feature matching. Two images of a desk taken at different angles. The superposition shows the detected corner features and the corresponding matches. (Source: MATLAB documentation.)

3.4.3 Feature Tracking

It can be computationally expensive to calculate the optical flow for each pixel in an image. A more computationally efficient technique is to detect image features in an image and then track them using the LK method.

3.5 Motion Model

A motion model is used to describe the overall motion of the scene with a few degrees of freedom (DOF). Doing this makes the computation less expensive.

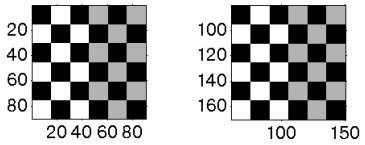
A motion model can then be represented by a transformation matrix:

$$H = \begin{pmatrix} a & c & 0 \\ b & d & 0 \\ dx & dy & 1 \end{pmatrix} \tag{3.11}$$

Depending on the DOF we can have different motion models to describe the motion from one frame to the next.

- **Translation** has 2 DOF. Translation in the x and y direction.

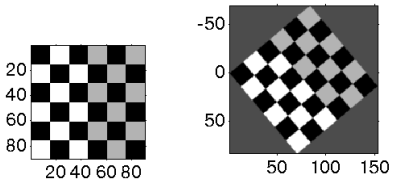
$$H = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ dx & dy & 1 \end{pmatrix} \tag{3.12}$$



(a) Original (b) Translation

- **Similarity** has 4 DOF. Translation, uniform scale and rotation.

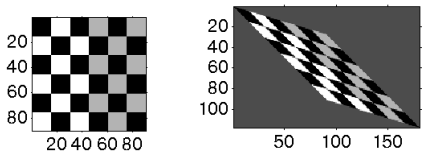
$$H = \begin{pmatrix} a & -b & 0 \\ b & a & 0 \\ dx & dy & 1 \end{pmatrix} \tag{3.13}$$



(c) Original (d) Similarity

- **Affine** has 6 DOF. Translation, rotation, independent scaling and shearing.

$$H = \begin{pmatrix} a & c & 0 \\ b & d & 0 \\ dx & dy & 1 \end{pmatrix} \tag{3.14}$$



(e) Original (f) Affine

Given the homogeneous coordinates of a point $(x_0, y_0, 1)^T$, the coordinates of the point after the transformation are obtained as $(x_1, y_1, 1) = (x_0, y_0, 1)H$.

3.6 RANSAC Algorithm

The RANSAC (RANdom SAmple Consensus) algorithm [4] is used for estimating the parameters of a model from a dataset containing many outliers. In the context of motion estimation, RANSAC can be used for feature matching or to exclude tracked features that do not fit the motion model.

A basic overview of the algorithm is as follows:

Data: M data points
Result: Estimated model and inlier points
initialization;
for L *times* **do**
  Select randomly N data points;
  Estimate the parameters of the model;
  Find how many of the M original points fit the estimated model given a specified tolerance ϵ. Call these inlier points.
  if *the number of inlier points exceeds a threshold* τ **then**
    Re-estimate the model using the inlier points;
    Terminate;
  end
end

Algorithm 1: RANSAC Algorithm

It should be noted that the outcome of RANSAC depends on the number of iterations L, the higher the number, the more chances that a better fitting model can be found.

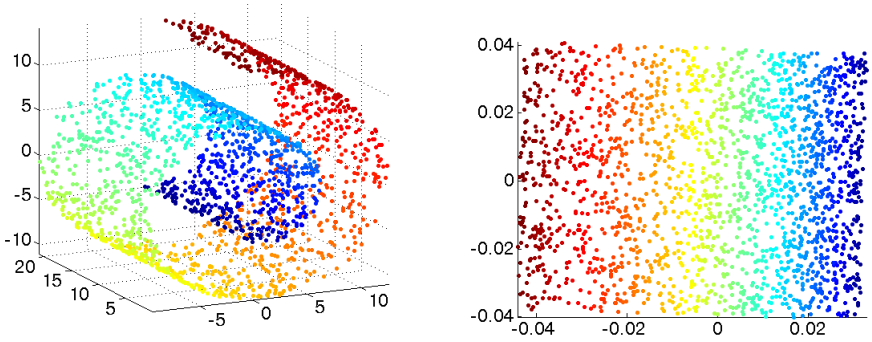
3.7 Manifold Learning

3.7.1 Overview

Manifold Learning is a set of non-linear dimensionality reduction techniques that aim to uncover a considerably lower dimensional manifold while preserving the structure of the high dimensional data. In the ideal case, the dimensionality of the uncovered manifold should be that of the intrinsic dimensionality of the data. There exists a couple of powerful techniques that have been proposed such as, Isomap [23] and Laplacian Eigenmaps [24], among others. In the context of medical image analysis, the use of such techniques offers an appealing solution as images could typically consist of millions of data points, so applying manifold learning can reduce the dimensionality of the image by uncovering a lower dimensional embedding and facilitate the application of subsequent processing. The powerful advantages offered by manifold learning has led to a considerable interest both in machine learning and computer vision communities. Manifold learning has found many successful application [25] such as pose estimation, video content analysis, image segmentation and object tracking [26] based on dynamical models.

One interesting application of manifold learning of relevance to this project is the possibility of being able to learn the cardiac and respiratory cycles from the video. Previous works have demonstrated this [27], [28]. Laplacian Eigenmaps have been used in a hierarchical setting to learn the different cardiac and respiratory motion occurring in real-time cardiac MRI sequence [29].

According to a survey [25] about manifold learning for images, a major limitation for manifold learning is that methods usually define a mapping from the original space to the lower dimensional one and not in the other way around. The inverse projection is still a challenge both theoretically and practically for non-linear manifold learning. Some methods have been proposed to compute the inverse but usually most methods do so by finding the nearest neighbours in the high dimensional



(g) 3D input data arranged in a swiss roll (h) Low dimensional embedding

Figure 3.7: Manifold learning example, we see that even thought the data in (a) are described by 3 coordinates, there exists a 2D embedding as in (b). The aim of manifold learning is to uncover such underlying embeddings. (Source: Created using Matlab)

space and then interpolating their respective embedded coordinates.

3.7.2 Laplacian Eigenmaps

Laplacian eigenmaps [24] employ spectral techniques to carry out non-linear dimensionality reduction. The technique aims to uncover a manifold that preserves local properties of the data. In essence it aims to keep data points that are locally near each other in the high dimensional space also locally near in the lower dimensional embedding.

Let $X = (x_1, ..., x_N)^T$, $x_i \in \mathbb{R}^D$ be the input data points in the high dimensional space and $Y = (y_1, ..., y_N)^T$, $y_i \in \mathbb{R}^d$ the projection of these points on the lower dimensional manifold, where $d \ll D$. In the case of images, x_i would be a vector of intensities and D the number of pixels or the number of features extracted from the image.

To capture the local similarity, an undirected graph can be constructed by using n nearest neighbours i.e. there is an edge between x_i and x_j if x_i is one of the n nearest neighbours of x_j according to some distance measure. A weight W_{ij} is associated to the edge connecting x_i to x_j.

A heat kernel can be used for the weights

$$W_{ij} = \begin{cases} e^{-\frac{||x_i - x_j||^2}{t}} & \text{if i and j are connected} \\ 0 & \text{otherwise} \end{cases}$$

or simply

$$W_{ij} = \begin{cases} 1 & \text{if i and j are connected} \\ 0 & \text{otherwise} \end{cases}$$

The graph Laplacian is then defined as $L = D - W$ where D is a diagonal matrix with $D_{ii} = \sum_j W_{ij}$
The embedding is then found by minimising the following

$$f(Y) = \sum_{ij} ||y_i - y_j||^2 W_{ij} = 2Y^T LY \tag{3.15}$$

subject to the constraint $Y^T DY = 1$ which is needed in order to remove an arbitrary scaling factor. The solution to 3.15 is given by the eigenvectors y corresponding to the lowest non zero eigenvalues of the generalised eigenvalue problem $Ly = \lambda Dy$.

3.8 Linear Programming

Given a linear function $f(u_1, ..., u_n)$, the aim of linear programming is to find the values of the the variables u_i, while being subject to a set of constraints, such that the value of the objective function f is minimised (or maximised). Any point that satisfies the constraints is called a feasible point. Linear programming offers a way to find the best feasible points such that the objective function is minimised (or maximised).

A linear program is an optimisation problem typically of the form:

$$\begin{aligned} &\text{Minimise } \mathbf{c}^T\mathbf{x} \\ &\text{subject to } A\mathbf{x} = \mathbf{b} \\ &\mathbf{x} \geqslant 0, \end{aligned}$$

where $\mathbf{c} \in \mathbb{R}^n$, $\mathbf{b} \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m \times n}$. Different variations of the above problem are possible.

3.9 Summary

In this chapter we have looked at what image features are, the notion of the motion field and its approximation using optical flow. We have seen that there are different types of image feature detectors and looked at methods by which we can track them across image sequences. We are aiming to use these so as to be able to obtain a motion model that would represent the overall motion in the scene between two consecutive frames which is estimated using a statistical algorithm such as RANSAC.

Finally we looked at an algorithm for non-linear dimensionality reduction, of which the use will be more clear in the next chapter.

Chapter 4

Design / Theory

This chapter provides the methods and theories that go into the motion stabilisation techniques. We also look at a specular highlights detection and in-painting procedure.

4.1 Video Stabilisation Overview

Before proceeding we remind the steps that go into video stabilisation:

1. **Original motion estimation:** This step consists of obtaining an estimate of the original camera path from the video using a motion model that describes the overall original motion.
2. **New path estimation:** This consists in estimating a potentially smooth camera path from the estimated original camera path.
3. **Synthesising / Re-rending:** A new video is recreated from the smoothed path viewpoint.

4.2 Motion Estimation

Motion estimation is done by first estimating a sparse motion field. To do this, we first detect features (Shi-Tomsai, FAST or SURF) in the frame under consideration and track them into the next frame using the pyramidal Lucas-Kanade (LK) differential tracking algorithm [21]. We could also make use of SURF features matching technique as a possible feature tracking alternative. After having estimated the motion field in the form of a sparse optical flow, the overall motion between each pair of consecutive frames is described as a geometric transformation (the motion model).

Motion estimation can be done **globally** i.e. performing tracking on the entire scene and using the location of all the tracked points to estimate the motion model, or **locally** i.e. performing tracking on a specific fixed region or using an object tracker. In the the case of local and/or object tracking, if the area being tracked is relatively small then a translation model would be better suited, as the smaller the area is, the fewer tracked point we have to get a good estimate for the model if it has a high DOF.

4.2.1 Tracking Using the Lucas-Kanade Method

Tracking is first done by selecting a set of points that could be easily tracked. The technique usually performs well for tracking elements that do not change shape and/or if they have a visual texture that makes it easy to detect features on them.

The tracker can be combined with different image feature detectors such as those mentioned in the previous section. Points are reacquired periodically at each frame so that there is always a consistent amount of tracked points.

In this project we use the tracker with FAST or SURF features in a global motion estimation setting. We use the SURF features predominately with the medical video sequences as it has been shown to perform best for tracking cardiac sequences [14].

Outlier Rejection

Typically outliers are detected by using error measures such as the sum of squared difference SSD or the normalised cross correlation NNC to compare the local neighbourhood around the estimated location of the tracked pixel. Points having an SSD and/or NNC measure above a certain threshold are considered as outliers.

4.2.2 Median Flow Tracker

A method for detecting tracking failures based on the Forward-Backward (FB) error is proposed here [30]. The tracker, called Median Flow, uses the pyramidal Lucas-Kanade method for tracking detected points forwards and backwards and eliminates points that do not agree within a certain threshold. This is justified by the fact that if a point is tracked well from the first frame to the second then tracking it back from the second to the first should make it come back to its original location.

Given two pairs of images I_t, I_{t+1} and a bounding box, the tracker works by first initialising points in a grid like fashion inside the bounding box. The points are then tracked using the LK tracker. To each one of the points is associated an error FB, NCC and the SSD. A point is then rejected if its FB error is greater than the median of the FB error of all the points. Similarly for NCC and SSD. The remaining inlier points are used to predict the new position of the bounding box in I_{t+1} by calculating the median in the x and y directions. The scale of the bounding box is also changed and is computed by calculating the ratio between the current distance and the previous distance for each point and is then taken as the median of these ratios. The underlying assumption for this is that the object is made of small rigid patches. Figure 4.1 shows a diagram of the tracking procedure. It should also be noted that object tracking can be sensitive to initialisation.

We use this tracker in this project in an object tracking setting as well as a global motion estimation setting. For the local object tracking setting we only estimate the motion according to a translational motion model. It is calculated as being the translation of the centre of the bounding box between consecutive frames. The proposed tracker has the size of the bounding box change according to the tracked features. However by testing it on scenes with deformable tissue, where there is relatively minimal depth variation, we noticed that the size of the bounding box tended in some areas to increase continually due to the periodic contractions, so in this project we modified it to so as it stays of the same constant size as in the initialisation.

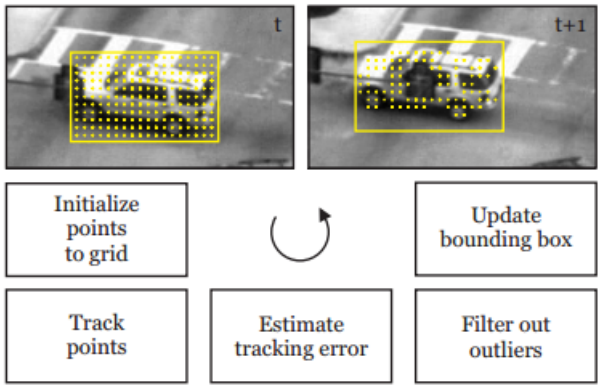


Figure 4.1: Diagram of Median Flow tracker. (Source: [30])

4.2.3 Tracking-Learning-Detection

Tracking Learning Detection (TLD) is an adaptive tracking framework proposed by Kalal et al. [31] that aims at addressing long-term tracking of unknown objects in a video sequence. In this framework, tracking an object in a scene is decomposed into 3 tasks: tracking, learning and detection. Typical issues that arise in long term tracking are occlusions, object disappearing and reappearing later in a scene, so the addition of an object detector can allow reinitialisations. Also an object can change shape, so learning to detect the new shapes of the object is useful.

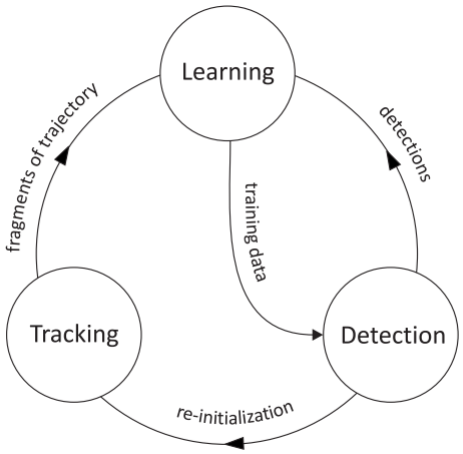


Figure 4.2: Block diagram of the TLD framework. (Source:[31])

We will briefly outline how the algorithm works without going much into details as the tracker is not the main object of the project and is only mentioned for consideration as a possible future direction to investigate for stabilisation.

The framework of the tracker can be seen in figure 4.2. The tracker estimates the motion of the object between consecutive frames. If the tracker fails, whether it drifts off the object or the object gets occluded or gets out of frame, then the tracker alone cannot recover. The detector considers each frame as a new independent frame and scans it entirely to locate the objects in the image that most resemble the description of the object. A detector usually makes errors. The learning step is used to train the detector based on the trajectory of the tracker.

For initialisation, the area/object that is desired to be tracked is selected with a bounding box. The image patch in the bounding box is used to build the initial object model in the form of features. The object model can grow and change over time as more images patches that represent the object are acquired in the learning step.

Subsequently, for each new frame:

- The area is tracked using the Median Flow tracker.

- The detector scans the entire image in the form of a scanning window and makes decisions whether the object is present or no. The number of image patches to be evaluated is large, so an initial step using integral images eliminates the patches that have a variance lesser than 50% that of the object (certain calculations can be achieved faster on image integrals). Comparisons are then made and the nearest matches to the object are returned with confidence values representing how close the matches are to the object.

- The final location of the object in the scene is obtained by the combination of both the position of the tracker and the detector. The position of the tracker can be initialised or re-initialised based on the confidence of the matches obtained from the detector and the confidence of the area being tracked by the Median Flow tracker.

A limitation of the tracker as identified by the authors is that the tracker does not perform well with full out-of-plane rotations and with articulated objects.

4.2.4 Estimating the Geometric Transformation

The geometric transformation between two consecutive frames is estimated using the RANSAC algorithm [4] by using the tracked feature points as input.

When performing tracking on a small area, we use a translation model as given the small amount of good features that can be detected, it is not possible to obtain a good estimate if the model has higher DOF. We estimate the translation using the method as described for the Median Flow tracker.

4.3 Camera Path

To perform stabilisation we need to estimate the motion of the camera path $C(t)$ as a function of time. Assuming it is obtained with a motion estimation technique, it can be defined at each frame in discretised form as

$$C_{t+1} = C_t F_t \implies C_t = F_1 ... F_t \tag{4.1}$$

F_t is the transformation matrix and it describes the motion model from frame I_t to I_{t-1}.

4.4 Zero Motion Stabilisation

A straight forward way to stabilise a video sequence to have zero motion is simply applying the inverse of the original camera path C_t at frame t to get the zero motion P_t:

$$P_t = C_t C_t^{-1} = \mathbf{I} \tag{4.2}$$

This works well assuming that we have a really good estimate of the camera path otherwise errors would eventually accumulate. Another issue is that if the entire scene from the initial frame goes out of frame then the stabilised video would also go out of frame. Which is why the technique described in the next section is better suited in most cases.

4.5 Stabilisation with L1 Optimal Camera Paths

The goal of the stabilisation technique by Grundmann et al. [1] is that given an original shaky camera path $C(t)$, to compute a stabilised camera path $P(t)$ having characteristics as if it were shot from a cinematic point of view, which is usually conveyed by the use of static, panning and dolly shots. To achieve this, the problem is formulated as a constrained linear program to minimise the first, second and third derivatives of the desired optimal camera path $P(t)$. The techniques finds optimal path segments of either static, linear or parabolic motion and avoids the superposition of the three.

Each derivative minimised represents a different type of path:

- The first derivative: $DP(t) = 0$. A constant path.
- The second derivative: $D^2P(t) = 0$. A constant velocity path.
- The third derivative: $D^3P(t) = 0$. A constant acceleration path.

This is achieved by considering a crop window of a fixed aspect ratio and then moving it along a path that is optimised to have the above properties while being subject to various constraints, such as, an inclusion constraint so as to keep the crop window constantly within the bounds of the frame.

Assuming the original camera path $C(t)$ is obtained by motion estimation, we would have

$$C_{t+1} = C_t F_t \Longrightarrow C_t = F_1 ... F_t \tag{4.3}$$

F_t represents the motion model from frame I_t to I_{t-1}. Given this path, the stabilised path can be expressed as

$$P_t = C_t B_t \tag{4.4}$$

where B_t represents the update transform that is applied to the original path to obtain the smooth stabilised path. The aim of the stabilisation algorithm is to find the update transform B_t for each frame. We have

$$B_t = \begin{pmatrix} a_t & c_t & 0 \\ b_t & d_t & 0 \\ dx_t & dy_t & 1 \end{pmatrix} \tag{4.5}$$

since it is also a geometric transformation matrix. So for a point $x = (x_1, x_2)$, we write applying the update B_t to x in the following form

$$A(x, p_t) = \begin{pmatrix} x_1 & x_2 & 1 \end{pmatrix} \begin{pmatrix} a_t & c_t \\ b_t & d_t \\ dx_t & dy_t \end{pmatrix} \tag{4.6}$$

where $p_t = (dx_t, dy_t, a_t, b_t, c_t, d_t)^T$ is a parametrisation vector of B_t.

4.5.1 Objective Function

The optimal path P_t is computed by minimising the following objective function:

$$f(P) = w_1 \left|D(P)\right|_1 + w_2 \left|D^2(P)\right|_1 + w_3 \left|D(P)^3\right|_1 \tag{4.7}$$

subject to constraints which will be mentioned in the next subsection. The weights w_1, w_2 and w_3 define the relative importance of each term.

Minimising $|D(P)|_1$ By using forward differencing $|D(P)| = \sum_t |P_{t+1} - P_t|$ and by using the substitutions from 4.3 and 4.4 we get:

$$\begin{aligned} |D(P)| &= \sum_t |C_{t+1}B_{t+1} - C_tB_t| \\ &= \sum_t |C_tF_{t+1}B_{t+1} - C_tB_t| \\ &\leq \sum_t |C_t|\,|F_{t+1}B_{t+1} - B_t| \\ &\leq \sum_t |C_t|\,|R_t| \end{aligned}$$

where $R_t = |F_{t+1}B_{t+1} - B_t|$. Given that C_t is known, we therefore want to minimise the residual

$$\sum_t |R_t| \tag{4.8}$$

To make it more suited for the problem formulation we rewrite R_t in parameter form

$$R_t(p) = M(F_{t+1})p_{t+1} - p_t \tag{4.9}$$

where $M(F_{t+1})$ is a linear operation transforming F_{t+1} such that $M(F_{t+1})p_{t+1}$ would represent the matrix multiplication $F_{t+1}B_{t+1}$ in vector form.

Minimising $\left|D^2(P)\right|_1$ Forward differencing yields $\left|D^2(P)\right| = \sum_t |DP_{t+2} - DP_{t+1}|$
$= \sum_t |P_{t+2} - 2P_{t+1} + P_t|$. However to model the error as additive, the difference of residual is minimised directly instead

$$\sum_t |R_{t+1} - R_t| = \sum_t |R_{t+1}(p) - R_t(p)| \tag{4.10}$$

Minimising $\left|D^3(P)\right|_1$ Similarly we want to minimise:

$$\sum_t |R_{t+2} - 2R_{t+1} + R_t| = \sum_t |R_{t+2}(p) - 2R_{t+1}(p) + R_t(p)| \tag{4.11}$$

Linear Programming (LP) To minimise the L1-norm of each of the residuals, slack variables are introduced. Each one of the residuals will require the introduction of N slack variables, where N is the dimension of the parametrisation (e.g. $N = 6$ for an affine motion model). Given that there are 3 types of residuals, for n frames, a total of $3nN$ slack variables have to be introduced.

e.g. for $|DP|$ we have

$$-e_t^1 \leq R_t(p) \leq e_t^1 \tag{4.12}$$

with $e_t^1 \geqslant 0$ a vector of length N. So the objective is to minimise

$$c^T e^1 = \sum_{t=1\ldots n} c^T e_t^1 \tag{4.13}$$

where $e^1 = (e_1^1, \ldots, e_n^1)$. If $c = \mathbf{1}$, this would correspond to the minimisation of the L1-norm. However since the translation part of p_t is at a different scale than the strictly affine part, it is necessary to weight the latter higher, e.g. for $N = 6$, $c^T = (1, 1, s, s, s, s)$, where s is the weighting. It is suggested in [1] to choose $s = 100$.

Similarly for $|D^2P|$ and $|D^3P|$.

The Final Objective Function is obtained by combining the three functions:

$$c^T(w_1 e^1 + w_2 e^2 + w_3 e^3) \tag{4.14}$$

with w_1, w_2 and w_3 being scalar weights associated to each one of the derivative constraints as in equation 4.7.

4.5.2 Constraints

Proximity Constraints To avoid having excessive changes in scale and rotation which might result from the chance that it could minimising the residuals better, hard constraints are imposed to limit the amount they can deviate:

$$0.9 \leq a_t, d_t \leq 1.1, -0.1 \leq b_t, c_t \leq 0.1, -0.05 \leq b_t + c_t \leq 0.05 \text{ and } -0.1 \leq a_t - d_t \leq 0.1 \tag{4.15}$$

Making use of an upper and lower bounds reformulation, the above can be rewritten as

$$lb \leq Up_t \leq ub \tag{4.16}$$

where U represents the linear combination over p_t such that we would have the relations in equation 4.15 be represented in matrix form.

Inclusion Constraint This constraint is necessary to insure that the crop window does not go out of frame when it is transformed by the update transform. For this to be satisfied, the 4 corners of the crop window $c_i = (c_i^x, c_i^y)$ $i = 1..4$ have to stay within the frame of height h and width w.

By posing $CR_i = \begin{pmatrix} 1 & 0 & c_i^x & c_i^y & 0 & 0 \\ 0 & 1 & 0 & 0 & c_i^x & c_i^y \end{pmatrix}$, we require:

$$\begin{pmatrix} 0 \\ 0 \end{pmatrix} \leq CR_i p_t \leq \begin{pmatrix} w \\ h \end{pmatrix} \tag{4.17}$$

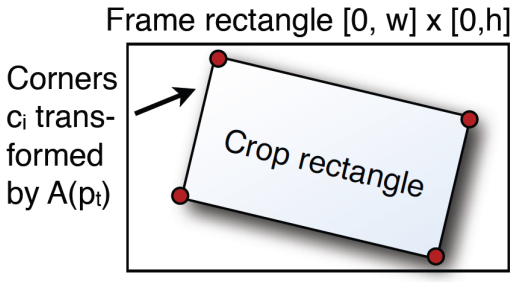


Figure 4.3: Inclusion constraint. (Credit: [1])

4.5.3 Additional Saliency Constraints

The formulation of the problem easily allows the introduction of saliency constraints, that is, it is possible to specify that we want a given point or region at a specific frame to remain within the crop window. Let s_i^t be the set of salient points at frame I_t, we can introduce the constraint of how far at least the salient point should lie from a given point (b_x, b_y)

$$\begin{pmatrix} 1 & 0 & s_i^x & s_i^y & 0 & 0 \\ 0 & 1 & 0 & 0 & s_i^x & s_i^y \end{pmatrix} p_t - \begin{pmatrix} b_x \\ b_y \end{pmatrix} \geqslant \begin{pmatrix} -\epsilon_x \\ -\epsilon_y \end{pmatrix} \tag{4.18}$$

$\epsilon_x \geqslant 0$ $\epsilon_y \geqslant 0$ are new slack variables to be added to the LP. We could then introduce for each point in s_i^t a set of 4 bounds (b_x, b_y) that would correspond to the 4 corners of the crop window.

It is best if ϵ_x and ϵ_y are non-zero, otherwise they might conflict with the inclusion constraint and disrupt the smoothness of the path.

4.5.4 Summary for the L1 Optimal Camera Paths

Overview

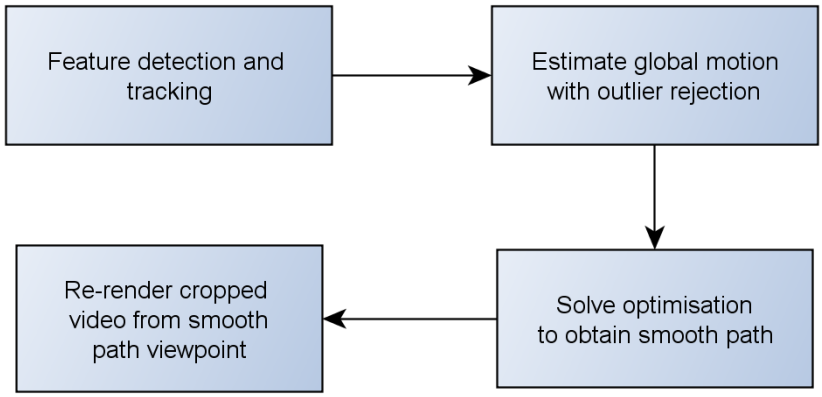


Figure 4.4: L1 optimal paths stabilisation algorithm main steps overview.

LP

Input: Frame pair transforms, $F_t, t = 1..n$
Output: Optimal camera path $P_t = C_t B_t = C_t A(p_t)$

$$
\begin{aligned}
&\text{Minimise } c^T(w_1 e^1 + w_2 e^2 + w_3 e^3) \\
&\quad \text{w.r.t. } p = (p_1, ..., p_n) \\
&\quad \text{where } e^i = (e^i_1, ..., e^i_n), i = 1, 2, 3 \\
&\text{subject to} \\
&\text{smoothness } \begin{cases} -e^1_t \le R_t(p) \le e^1_t \\ -e^2_t \le R_{t+1}(p) - R_t(p) \le e^2_t \\ -e^3_t \le R_{t+2}(p) - 2R_{t+1}(p) + R_t(p) \le e^3_t \\ e^i_t \ge 0 \end{cases} \\
&\text{proximity } \{\ lb \le U p_t \le ub \\
&\text{inclusion } \{\ (0,0)^T \le C R_k p_t \le (w,h)^T, k = 1, 2, 3, 4
\end{aligned}
$$

4.5.5 Video Re-rendering

After having solved the optimisation as per the above described algorithm, the computed update transforms are applied to the crop window at each frame and is cropped to end up with a final stabilised video.

We noticed that it is possible in some cases to maximise the size of the crop window so as to keep as much of the image as possible, as it might happen that if we had chosen a bigger crop window than necessary for the optimisation some parts that could stay within the frame could get cropped even if they do not go out of frame. The maximised crop window is found by applying the update transform to each of the four corners of the of the original image and finding their amount

of displacement for the entire video, e.g. for the x^c_{left} coordinate of the left side of the maximised crop window

$$x^c_{left} = min(x_1^{topleft}, \ldots, x_n^{topleft}, x_1^{bottomleft}, \ldots, x_n^{bottomleft})$$

where $x_i^{topleft}$ is the x coordinate of the top left corner at frame i after applying the corresponding update transform to it.

4.6 Stabilisation for Scenes with Periodic Motion

In this section we look at two things. First, at how we could possibly use the L1 optimal paths stabilisation in a predictive scheme. Second, a possible alternative stabilisation technique based on manifold learning.

The stabilisation algorithm in the previous section is a post-processing algorithm, that is, we have to give it the entire video as input for it to work. This is much expected as the motion it is supposed to be dealing with is unpredictable. However in the case of the some of the medical image sequences, the motion that we desire to eliminate is that of the cardiac and respiratory motion which is periodic, hence it is feasible to assume that given that the motion is periodic, the stabilisation applied is bound to be periodic and repetitive. Solving the stabilisation for the entirety of a periodic sequence seems unnecessary. However, this approach means that we are making the assumption that the camera used to obtain the video is static, which unfortunately might not always be the case, a moving camera would introduce a different type of apparent motion in the scene. Also we assume that no new elements will be introduced in the scene to considerably affect it.

In essence, to perform predictive stabilisation, what we could envision to do is solve the stabilisation using the algorithm for an initial part of the video in which we see the periodic cycles a sufficient number of times and then for subsequent frames, depending on where the motion is in the cardiac and respiratory cycle, reapply the corresponding precomputed update transform.

4.6.1 Predictive Stabilisation Using K-Nearest Neighbour (K-NN)

We first perform motion estimation and stabilisation for an initial set of frames $1..N$ which we call training set, so we would have a set of images $I = I_1, ..., I_N$ and their corresponding set of update transforms $B = B_1, ..., B_N$.

When we encounter a new frame I_t with $t > N$, we look for its closest match / nearest neighbour in the training set and reuse its corresponding update. However there can usually be multiple close matches / nearest neighbours that have different or similar update transforms due to various errors (e.g. in the motion estimation), so the idea is to retrieve the K nearest neighbours and use them to estimate the new update transform.

The reason why we opt to look for the closest image match is that although it seems computationally extremely efficient to just look for the closest image position, motion estimation is typically prone to errors over time.

The predictive stabilisation would work as follows:

For a new frame I_t $t > N$ we find its K nearest neighbours in the set I using the euclidean distance and retrieve the corresponding K update transforms. We denote J as the set containing the indices of the K nearest neighbours. The update transform for the new frame can be obtained as a weighted interpolation

$$B_t = \frac{\sum_{i \in J} w_i B_i}{\sum_{i \in J} w_i} \tag{4.19}$$

The weight w_i can be chosen as a function of the distance between I_i and I_t.

$$w_i = \frac{1}{(||I_t - I_i|| + \alpha)^2} \tag{4.20}$$
$$\text{or} \quad w_i = e^{\frac{-||I_t - I_i||^2}{\sigma^2}}$$

where $\alpha > 0$ is needed to avoid dividing by zero. Using weighting based on the distance allows to reduce the influence of distant matches.

Limitations

This technique is computationally expensive as we would be looking for the closest image matches by comparing the distance between each pixel. We could rescale the image to a smaller size to reduce the computational cost at the price of affecting the performance.

Also the technique would fail if the camera moves or if there is a new type of motion or objects that are introduced in the scene. However the method seems like the most straight forward approach that we could initially try.

4.6.2 Stabilisation Using Manifold Learning

We explore in this section a technique that could be used to perform stabilisation based on manifold learning.

In [29], when using Hierarchical Laplacian Eigenmaps to reduce MR images to two dimensions, one embedded direction is found to be highly correlated to the heart cycle while the other being highly correlated with the respiratory cycle. Given that the motion in the x and y direction should depend on the position in the cardiac and respiratory cycle, we explore whether it is possible to find a direct representation of the x and y coordinates as a linear function of the embedded coordinates.

Motion Estimation / Trajectory Reconstruction Using the Embedded Coordinates

Let $I = (I_1, ..., I_N)^T$ be the set of input images $I_i \in \mathbb{R}^D$ represented in row vector form and $\mathbf{E} = (E_1, ..., E_N)^T$, $E_t \in \mathbb{R}^d$ the corresponding projection of I on the lower dimensional manifold, where $d \ll D$. $\mathbf{E}$ is a matrix of N rows and d columns.

We perform tracking using a translation motion model on an initial set of frames $1..N$ and obtain an estimated motion in the x and y direction, we then aim to find coefficients such that each coordinate can be expressed as a linear function of the embedded coordinates, i.e. for an image at frame t we aim to find an expression for the coordinates x_t and y_t in the form of

$$x_t = a_0^x + \sum_{i=1...d} a_i^x E_{ti} \tag{4.21}$$
$$y_t = a_0^y + \sum_{i=1...d} a_i^y E_{ti} \tag{4.22}$$

with a_i^x and a_i^y $i = 0 \dots d$ the coefficients. In matrix form, the above equations can be written as

$$X = \begin{bmatrix} \mathbf{E} & \mathbf{1} \end{bmatrix} \mathbf{a^x} \tag{4.23}$$
$$Y = \begin{bmatrix} \mathbf{E} & \mathbf{1} \end{bmatrix} \mathbf{a^y} \tag{4.24}$$

with $X = \left(x_1, \ldots, x_N\right)^T$, $Y = \left(y_1, \ldots, y_N\right)^T$, $\mathbf{a^x} = \left(a_0^x, a_1^x, \ldots, a_d^x\right)^T$ and $\mathbf{a^y} = \left(a_0^y, a_1^y, \ldots, a_d^y\right)^T$.

The overdetermined system can then be solved with least squares regression. An algebraic solution can be written as

$$\mathbf{a^x} = (\mathbf{E1}^T\mathbf{E1})^{-1}\mathbf{E1}^T X \tag{4.25}$$

$$\mathbf{a^y} = (\mathbf{E1}^T\mathbf{E1})^{-1}\mathbf{E1}^T Y \tag{4.26}$$

with $\mathbf{E1} = \begin{bmatrix}\mathbf{E} & \mathbf{1}\end{bmatrix}$.

Stabilisation

To obtain the stabilisation, we perform motion estimation using tracking on an initial set of frames in order to obtain an estimated x and y coordinates which are then used to perform regression to obtain an expression of the x and y position as a function of the embedded coordinates.

We then express this as a translation model for the camera path C_t at frame t as

$$C_t = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_t & y_t & 1 \end{pmatrix} \tag{4.27}$$

where x_t and y_t indicate the current position (and not the displacement between two consecutive frames) and are obtained with equations 4.21 and 4.22.

The stabilisation is then obtained by doing

$$P_t = C_t C_t^{-1} = \mathbf{I} \tag{4.28}$$

(as described in section 4.4).

Predictive Stabilisation

Using the above method it is also possible to attempt to perform predictive stabilisation using K-NN.

Motion estimation using a translation model is first performed for an initial set of frames $I = I_1, ..., I_N$ so as to be able to obtain an expression for the x and y coordinates as a function of the embedded coordinates.

Then, given a new image I_t $t > N$, we search for its K nearest neighbours in the set I and express its estimated embedded coordinates

$$E_t = \frac{\sum_{i \in J} w_i E_i}{\sum_{i \in J} w_i} \tag{4.29}$$

where J is the set of indices of the K nearest neighbours.

Finally we find an estimated x_t and y_t using equations 4.21 and 4.22.

Summary for Stabilisation

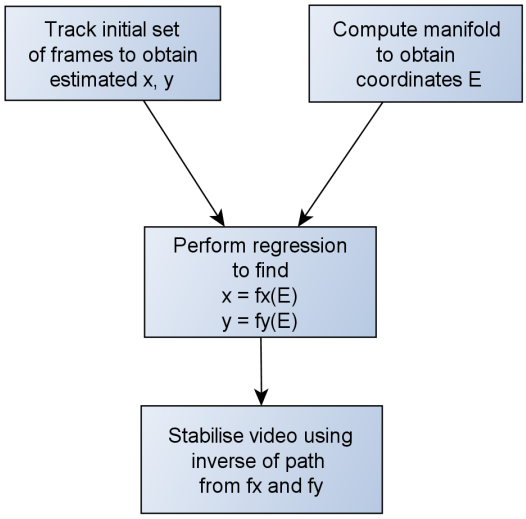


Figure 4.5: Stabilisation using manifold learning.

4.7 Specular Detection and Removal

The wet nature of the deformable tissue tend to cause a certain amount of specular highlights which in turn can introduce a considerable amount of error for the motion estimation algorithms. We look at two techniques for specular detection from [2] and [32] and use a modified specular detection and in-painting scheme mostly inspired from [2] by eliminating a few steps. We did not test whether we could achieve similar results as the method suggested in [2], but since we are mostly interested in finding the location of the specular areas to eliminate the tracked points that fall within it, the simplified scheme deemed sufficient.

4.7.1 Detection

In [32], they propose as a first step to convert the image from RGB to HSV (Hue, Saturation s, Value v) and then a pixel $\mathbf{p}$ is marked as being a specular highlight if

$$s(\mathbf{p}) < T_s,\ v(\mathbf{p}) > T_v \tag{4.30}$$

where T_s and T_v are thresholds, the values suggested in the paper are $T_s = 0.35$ and $T_v = 0.75$ (saturation and value vary from 0 to 1). For the next step, they segment the image into regions having similar texture and colour. However a disadvantage according to [2] is that the segmentation method they use is computationally expensive.

In [2], specular detection in the first step is done by using colour adaptive thresholds to find the bright highlights. Colour balancing can introduce intensity offsets in the colour channels, so they suggest normalising according to the ratios of the 95th percentile of their intensities to the 95th percentile of the grey/luma channel. The reason the grey intensity is used as opposed to the red channel is because reddish colors are very common in medical videos, so if the red saturates it cannot be only due to specular highlights.
With the luma channel being $c_L = 0.2989c_R + 0.5870c_G + 0.1140c_R$, with c_R, c_G and c_B the red, green and blue channel respectively, the ratios are computed as follows: $r_G = \dfrac{P_{95}(c_G)}{P_{95}(c_L)}$ and $r_B = \dfrac{P_{95}(c_B)}{P_{95}(c_L)}$, with $P_{95}(.)$ denoting the 95th percentile.

A pixel $\mathbf{p}$ is marked as a possible specular highlight if

$$c_G(\mathbf{p}) > r_G T_1 \ \vee\ c_B(\mathbf{p}) > r_B T_1 \ \vee\ c_L(\mathbf{p}) > T_1 \tag{4.31}$$

For the next step they propose applying a modified median filter over the image to find a representative colour of the neighbourhood of each pixel and then performing a comparison with the pixel. A pixel is considered as specular if the difference is greater than a certain threshold T_m. In this project we choose to simply perform this second step on just the grey channel. We therefore apply a median filter on the grey channel with a window size of $[sx, sy]$ and obtain the filtered image I_{median}. A pixel $\mathbf{p}$ is marked as a specular area if

$$c_L(\mathbf{p}) - I_{median}(\mathbf{p}) > T_m \tag{4.32}$$

We choose $T_m = 40$. For the median filter we choose $[sx, sy] = 50 \times 50$ as the videos used contain large specular areas.

The specular areas detected by equation 4.32 are more than those detected by equation 4.31. In the paper they proceed with additional steps based on the image gradient to remove falsely detected highlights, we however stop here.

4.7.2 In-painting

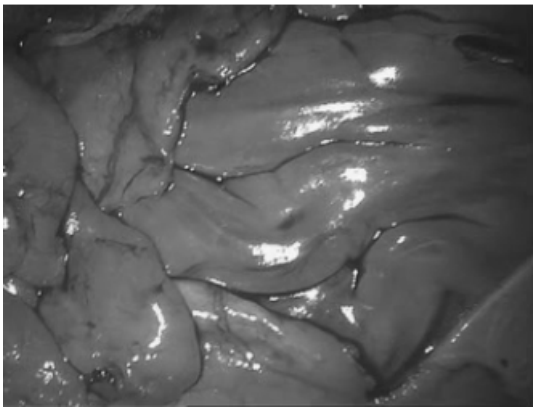
For the in-painting we use a technique inspired from [2]. The obtained binary $mask$ from equation 4.32 is dilated so as to avoid having visible borders around the detected areas. A Gaussian filter is then applied to it. Before applying it as a transition mask to blend the original image I_s with I_{median}, we add an additional step to avoid having the contrast of darker areas reduced which could happen if they are located right next to a specular area. We therefore detect darker areas with a threshold as follows

$$c_L(\mathbf{p}) < P_{50}(c_L) \tag{4.33}$$

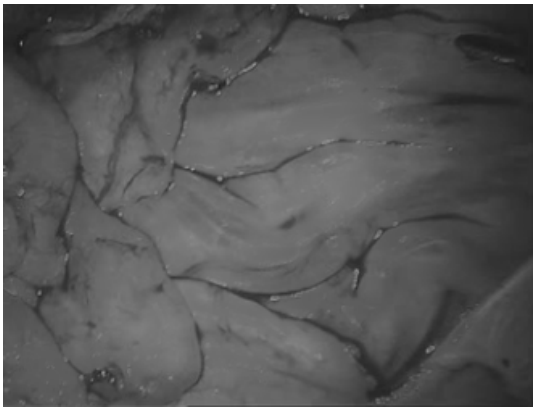
where $P_{50}(.)$ is 50th percentile. The obtained area is then filtered with a Gaussian filter to smooth the border and the resulting mask is removed from $mask$. The final image with the in-painting is obtain as

$$c_{ns}(\mathbf{p}) = (1 - mask(\mathbf{p})) \times c_L(\mathbf{p}) + mask(\mathbf{p}) \times I_{median}(\mathbf{p}) \tag{4.34}$$

which results in the specular area being filled with the value from the median filtered image. We can see in figure 4.6 the results of the specular highlight detection and in-painting.

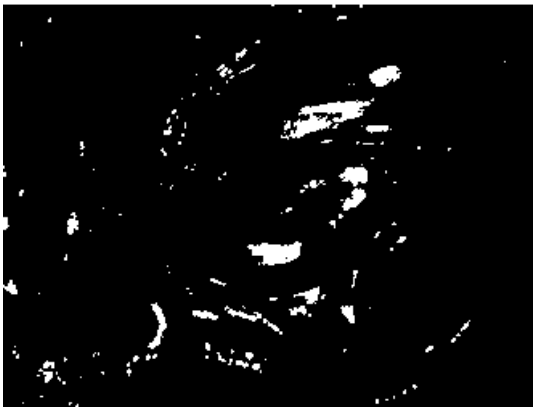


(a) Grey image of the heart having specular highlights



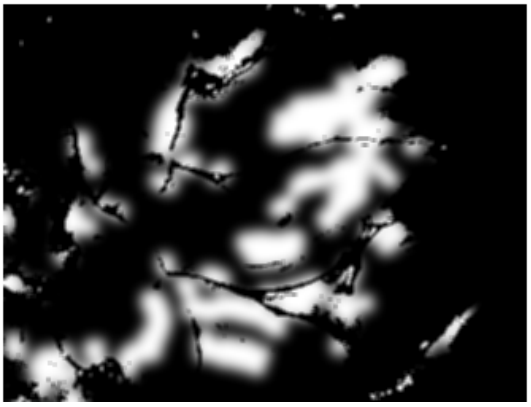
(b) In-painting

(c) Specular areas detected by equation 4.31

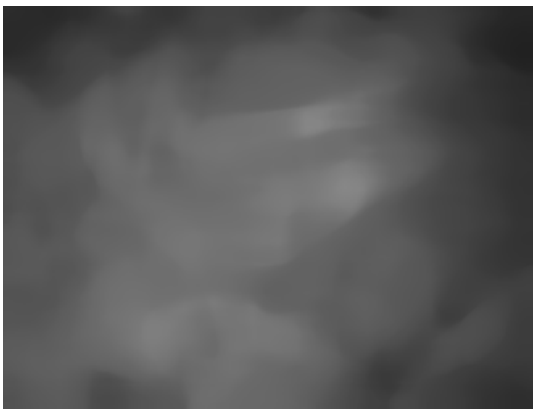


(d) Specular areas detected by equation 4.32

(e) Mask dilated and Gaussian filtered



(f) Darker areas removed from the mask



(g) Median filter size 30×30

(h) Median filter size 50×50

Figure 4.6: Specular highlight detection

4.8 Summary

In this chapter we have looked at the the theory behind the L1 optimal camera paths stabilisation technique. We suggested a possible naive scheme based on K-NN to use the update transforms obtained from the stabilisation algorithm to perform predictive stabilisation.

We have also looked at a possible stabilisation technique based on manifold learning whereby we seek to find an expression of the x and y coordinates as a linear function of the embedded coordinates.

In the last section of the chapter we looked at a specular highlights detection and in-painting method which should prove useful to have due to the nature of image sequences we are dealing with.

Chapter 5

Implementation

This chapters provides information about the software implementation.

5.1 Development Tools

Programming Language

The implementation was carried out mostly in MATLAB (R2013a) as it provides a quick way to start development and test out different things. This is mostly due to presence of different specialised toolboxes. MATLAB also offers the possibility of integrating C and C++ code through the use of MEX-files which are dynamically-linked subroutines that are loaded and executed by the MATLAB interpreter.

MATLAB Toolboxes

We predominately use the image processing and vision toolboxes in MATLAB. The vision toolbox provides the common computer vision functions such as feature detection, tracking and estimating the geometric transformation, among others.

Important note: The implementation was carried in MATLAB (R2013a) in which the vision toolbox has been slightly restructured, some function calls from the vision toolbox in MATLAB (R2013a) do not exist in the previous versions.

We also make use of OPTimization Interface (OPTI) Toolbox[1] which is a free MATLAB toolbox for solving linear, nonlinear, continuous and discrete optimisation problems. The toolbox has a range of open source and academic solvers. However the toolbox is delivered as an executable only for a Windows machine. To using it on different operating systems the source code needs to be compiled. We make use of this toolbox as it proved to be a faster solver than the existing solvers in MATLAB. In our implementation, if the toolbox is not installed then the MATLAB solver is used.

Other Technologies Used

OpenCV (Open Source Computer Vision)[2] is a library of computer vision C++ functions. It is used for the implementation of the Median Flow and TLD trackers mainly for the LK algorithm. It is integrated in MATLAB through the use of MEX-files.

GUI (graphical user interfaces)

The interface was created using MATLAB GUIDE (GUI development environment) which provides tools for designing user interfaces.

[1] http://www.i2c2.aut.ac.nz/Wiki/OPTI/index.php
[2] http://opencv.org/

Platform

The development was carried out on a Windows machine. However MATLAB code can be used on other platforms that have MATLAB installed although some external MEX files need to be recompiled.

5.2 User Interface

We created a graphical user interface so as to provide us with a means to better organise and view the results directly.

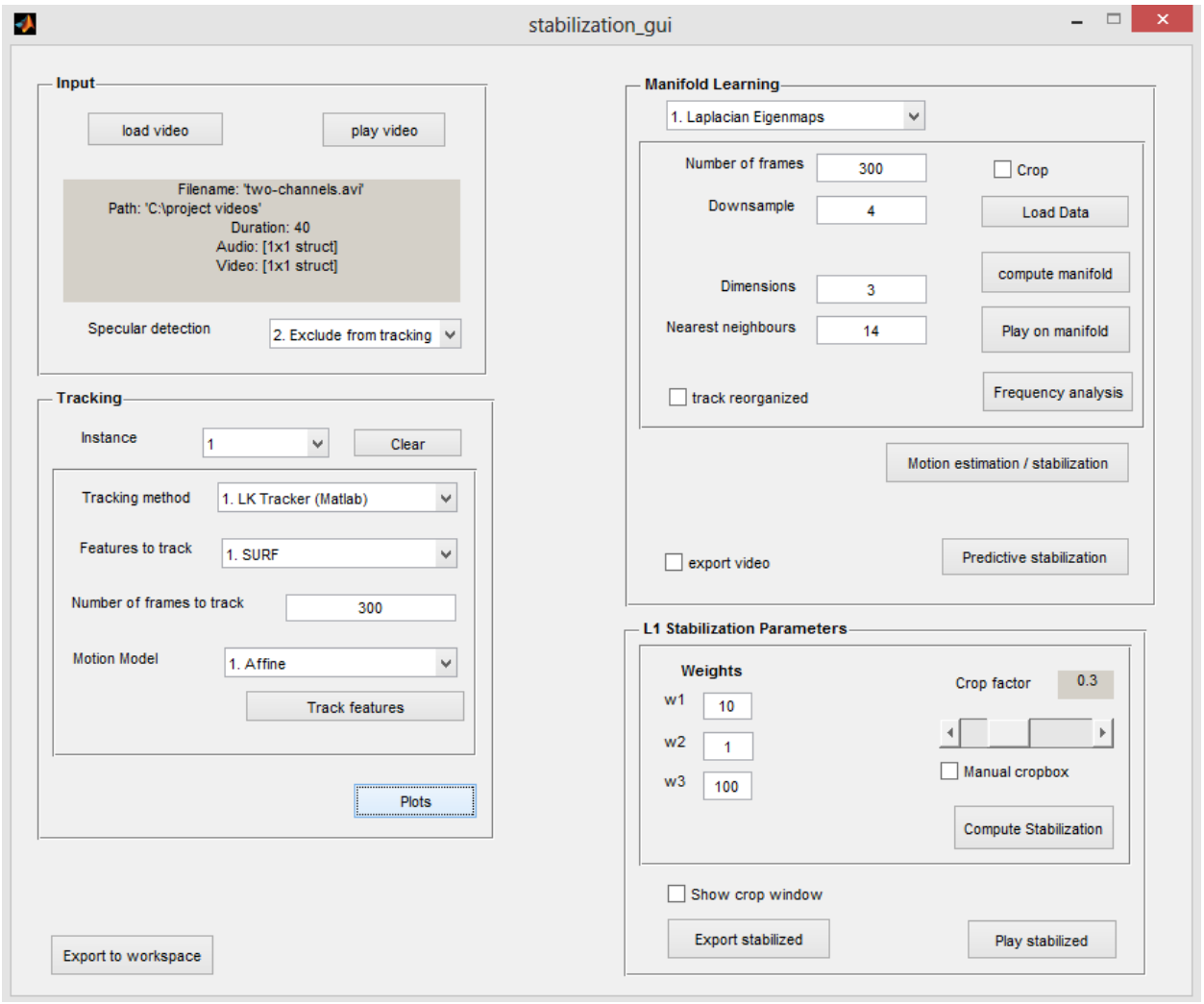


Figure 5.1: Main interface.

With the GUI, the user can...

Input related

- Load and playback the input video.
- The ability to exclude specular highlights from tracking or perform specular areas in-painting.

Tracking related

- Have the choice of selecting from 5 tracking methods and 3 different types of image features.

- view the results of tracking, the optical flow and the detected features on a video as it is being tracked.
- Generate plots so as to view the result of the estimated path.
- Have multiple tracking results stored in instances.
- Choose the number of frames to track.

Stabilisation related

- Select the size of the crop window manually or with a specified crop factor.
- Adjust the weight parameters.
- Stabilise the video and play the result.
- Plot the resulting smooth path.
- Export the stabilised video.

Manifold learning related

- Select between two manifold learning algorithms.
- Select the dimension and number of nearest neighbours for the manifold learning algorithm.
- Down-sample the image size with a factor.
- Different sets of plots related to the manifold.
- Play the video and see where each frame lies on the manifold with the evolution over time.
- Perform frequency analysis for the components.
- Use the results from manifold learning to perform motion estimation and stabilisation.

5.3 Tracking

LK Tracker

The vision toolbox in MATLAB provides the LK tracking algorithm. In our implementation we make use of it. We also make use of the LK implementation of the algorithm from OpenCV in MATLAB through the use of MEX-files.

TLD Tracker

The TLD[31] tracker is the result of the PhD thesis work of Zdenek Kalal, reimplementing the tracker would take a considerable amount of time hence we make use of the open source code that he provides[3].

We perform slight modification to the tracker to integrate it with our existing GUI. We also add to it a simple prediction scheme so as to see the effect of performance improvement that can be introduced. Although the algorithm can already achieve real-time performance we make this test to consider future possible work in incorporating a multi-tracking scheme to track multiple landmarks such that if tracking on a landmark fails then that of the other landmarks could still enable uninterrupted tracking.

[3]https://github.com/zk00006/OpenTLD

Chapter 6

Results and Evaluation I

In this chapter we present a set of experiments that we have performed and their corresponding evaluation.

In experiment 1, 2 and 3 we report the results of applying the stabilisation algorithm to 3 different videos, starting with a type of video that the stabilisation algorithm is intended for, for the second experiment we use a video of a synthetic heart and for the third one we use a video of the real heart.

6.1 Measurements

Before proceeding with the experiments, it would be beneficial to have some measurements that we could use to assess the performance of the stabilisation.

Mean Image can serve as a measure to show the effectiveness of a stabilisation algorithm. If there is a great amount of motion in the scene then the mean image would have a considerable amount of distortion and blur. However if there is no motion then there should not be a distortion as the images are all roughly the same and aligned.

The mean for N frames would be

$$I_{mean} = \frac{1}{N} \sum_{i=1...N} I_i \tag{6.1}$$

Modified Mean Image The stabilised video will end up being cropped, however we can still obtain a mean image that has the same size as the original video by performing a mean for pixel locations only when there is a value present. For a pixel p at a fixed location, we would have

$$I_{mean}(p) = \frac{1}{N(p)} \sum_{i \in J)} I_i(p) \tag{6.2}$$

where $N(p)$ indicates the number images in which there was a non-zero value present at the location of pixel p and J the set of indices of those $N(p)$ images.

Inter-frame Error serves to assess the effectiveness of the stabilisation by assessing how much the motion is reduced between consecutive frames. Large motion is expected to have larger inter-frame error. It can be obtained by calculating the mean squared error (MSE) between two consecutive frames.

For two consecutive frames I_t and I_{t+1} of size $W \times H$

$$e_{MSE}(I_t, I_{t+1}) = \frac{1}{W \times H} \sum_{i=1...H} \sum_{j=1...W} (I_t(i,j) - I_{t+1}(i,j))^2 \tag{6.3}$$

Although it has been shown that such objective pixel-wise comparison measures, such as the inter-frame error, do not usually correlate well with the perceived visual measurements [33] [34]. they are still commonly used, so we make use of them as they are easy to calculate and provide an instantaneous indicative measure of quality. Another type of possible measure is a subjective quality assessment such as a mean opinion score which consists of asking a set of observers to give their subjective opinion as to how good they think the stabilisation is, the downside however is that the process takes time. We do not make use of such measure.

6.2 Configurations

Another thing to note before proceeding with the experiments is that since there are different types of phases that go into the stabilisation and that each phase has its own set of configurations, testing each different possible configuration would be time consuming, hence they were set as per usually recommended in the literature.

Nonetheless, We have tried to change some of the configurations, such as, the type of features used, the number of detected features, the neighbourhood size for the tracker, the number of samplings and threshold for the the RANSAC algorithm, there were slight observed differences in the results but the overall conclusions remain the same.

6.3 Experiment 1: L1 Optimal Paths Stabilisation

6.3.1 Set-up and Results

In this section, we test the stabilisation algorithm on one of the original videos kindly provided by Grundmann, the author of the paper [1], to see how well our implementation performs.

The video

The video is of a person jumping while walking, the camera follows the person while having a considerable amount of shake, notably in the y direction. The 29fps video has a size of 640×360.

Motion Estimation

The tracking is performed on 300 frames. We detect FAST features and use the LK algorithm to track them. In figure 6.1 we have two frames taken from the video, we see that our outlier rejection using RANSAC performs as expected. Almost all the points detected on the person are correctly considered as outliers by the algorithm while almost all the points in the background are considered as inliers. We can see the sparse optical flow field in figure 6.2 on 4 frames, note how the background seems to have considerable amount of jittery movement, that the direction of the vectors in the background are similar in their local neighbourhood and finally that the directions of the vectors on the person are different than those belonging to the background.

Stabilisation

We then apply the stabilisation algorithm to the video by setting the crop window to 80% size that of the original frame. In the original paper [1], they used saliency constraints from a face detector to keep the person in the centre of the crop window, however here we do not do that.

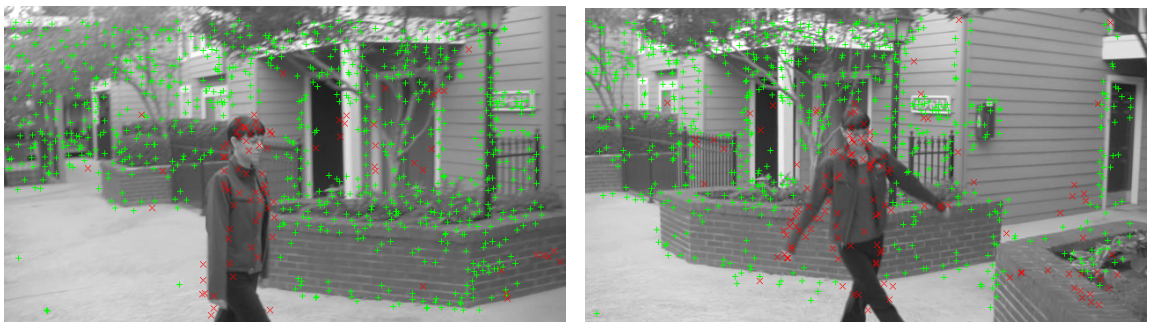
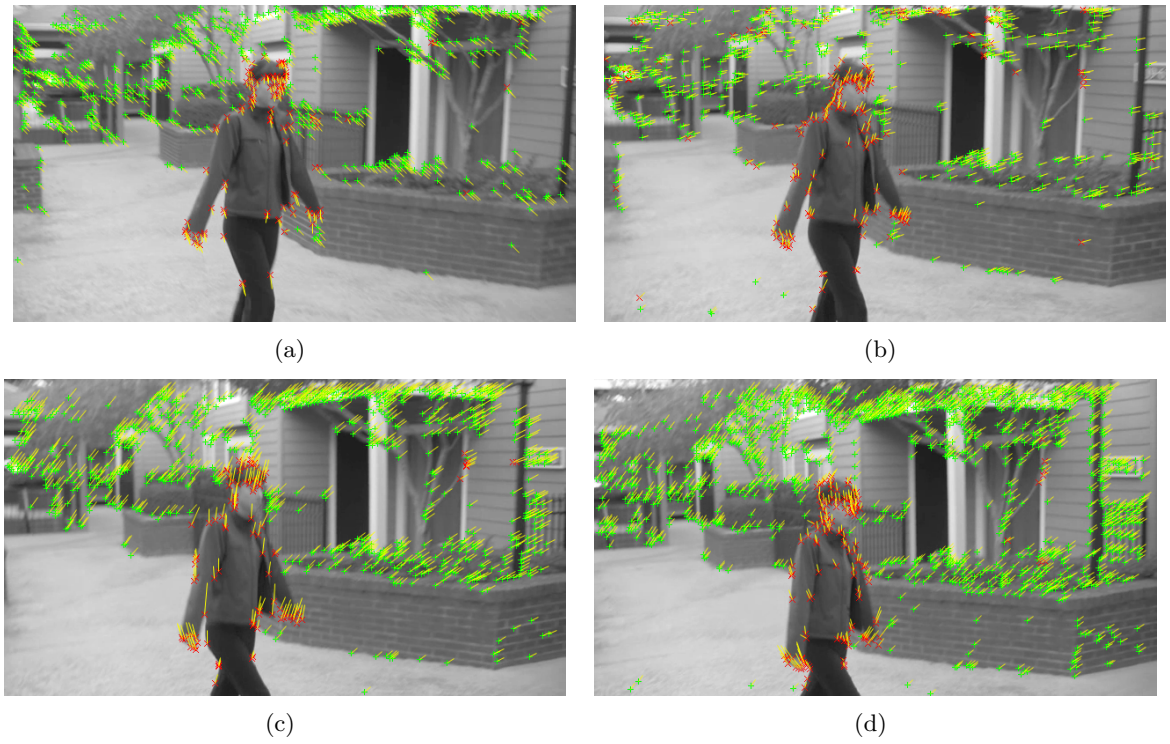


Figure 6.1: Two frames from a video with camera shake and a person jumping while waking. Red points are outliers and green points are inliers.

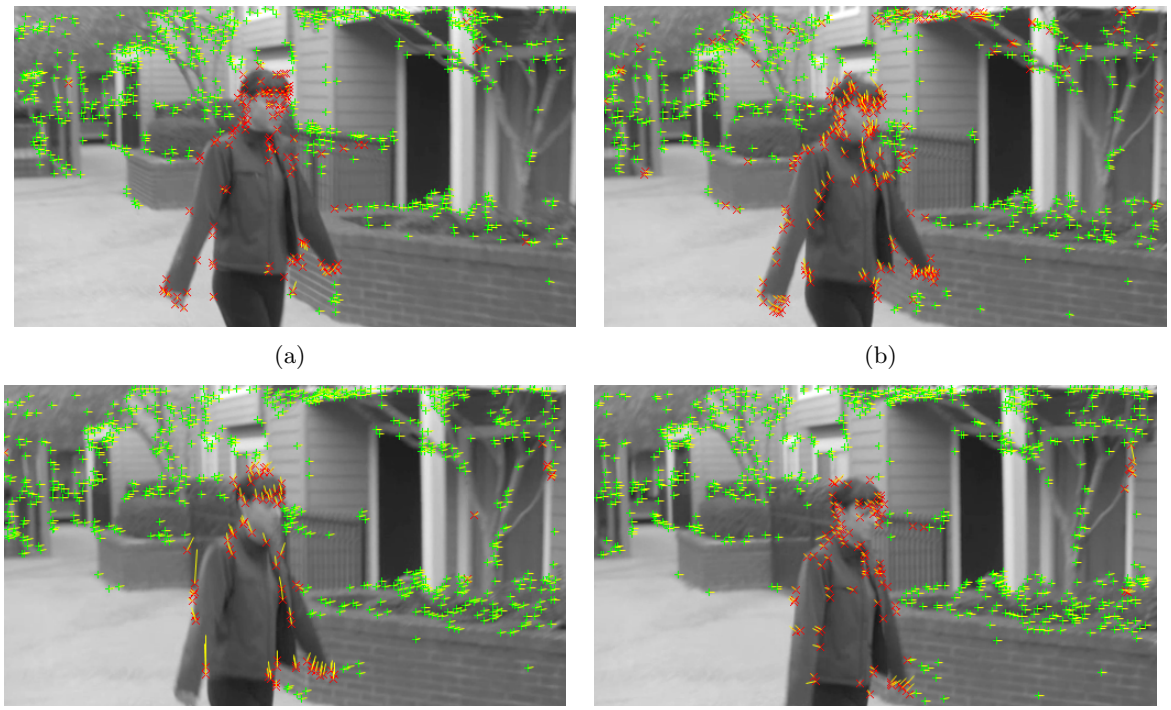


(a) (b)

(c) (d)

Figure 6.2: Four frames taken 2/29 seconds apart from the shaky video showing the optical flow, outlier points (red) and inlier points (green).

Figure 6.4 shows the original estimated camera path and the resulting smooth path from the L1 optimisation. We see that the smooth path tends to follow the direction of the original path while having no shake. Figure 6.5 shows the the position of the crop window on the original frame and the cropped stabilised result. We see that the stabilisation works as expected with the resulting video having a smooth camera path. Although we cannot apply here a pixel-wise objective measure to show the effectiveness of the stabilisation because the camera is moving, we can visually see the results by recomputing the optical flow field. We see that in figure 6.3, note how the background now has smoother motion which is exhibited by the constant direction and shortness of the vectors in the background.



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

Figure 6.3: Same four frames as in figure 6.2 taken from the stabilised video. Notice how the optical flow field in the background is smooth.

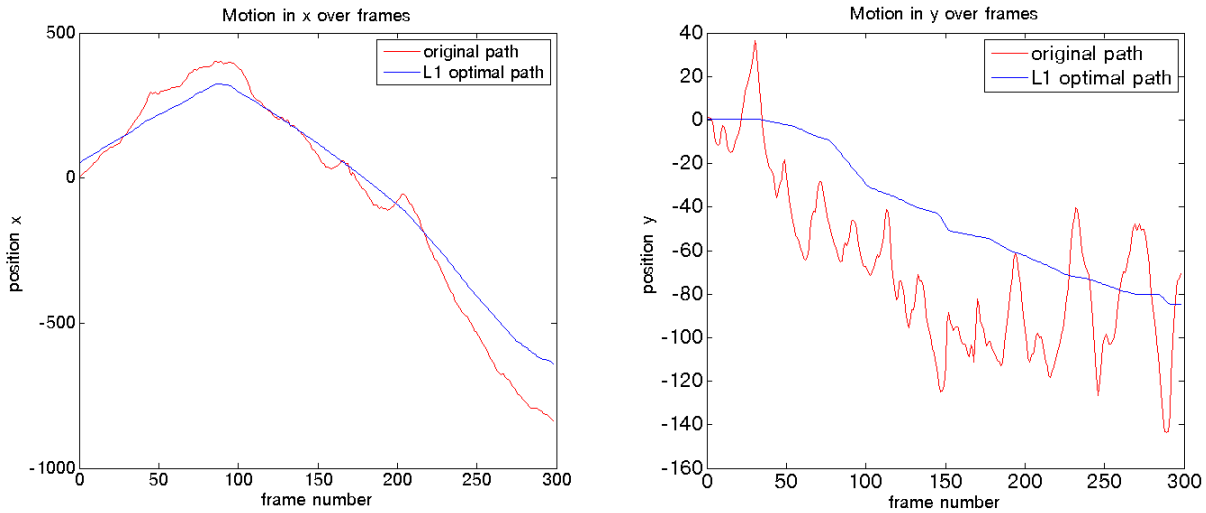


Figure 6.4: Graphs showing the estimated original camera path and the L1 optimal camera path. The crop window is 80% of the original frame. No saliency constraints have been used.

Figure 6.5: Four pairs of frames showing the results of the stabilisation. The red box on the frame above is the crop window on the original video and the the frame below is the cropped stabilised video.

6.3.2 Evaluation

We went through each step necessary for the stabilisation algorithm and applied it to a video containing a rigid background. We found that our implementation yields good results indicating that we have successfully implemented the algorithm.

6.4 Experiment 2: Phantom Heart

After having verified that the stabilisation algorithm works as expected, we now proceed to testing it on a video of a synthetic heart.

6.4.1 Set-up and Results

The video

The 25fps 360×288 video is of a silicon heart phantom [1]. The camera is static. The only visible motion is that of the periodic cardiac motion. There is no respiratory motion. The surface contains clearly visible landmarks. We use this video to see how well the stabilisation would perform in a controlled environment.

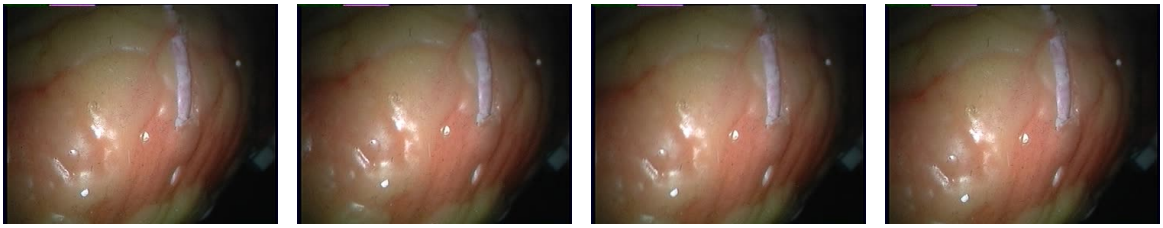


Figure 6.6: Four frames from the phantom heart video

Motion Estimation

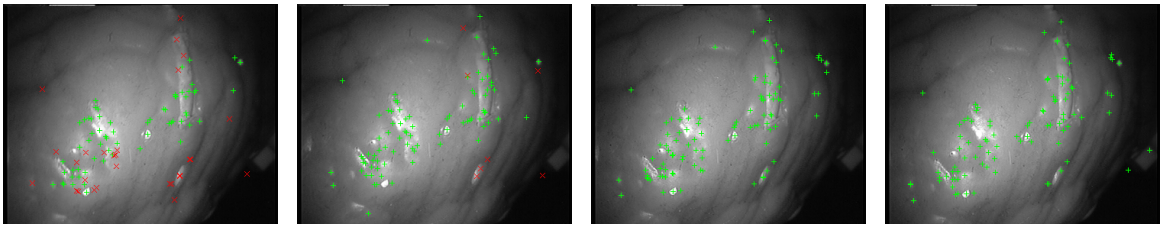
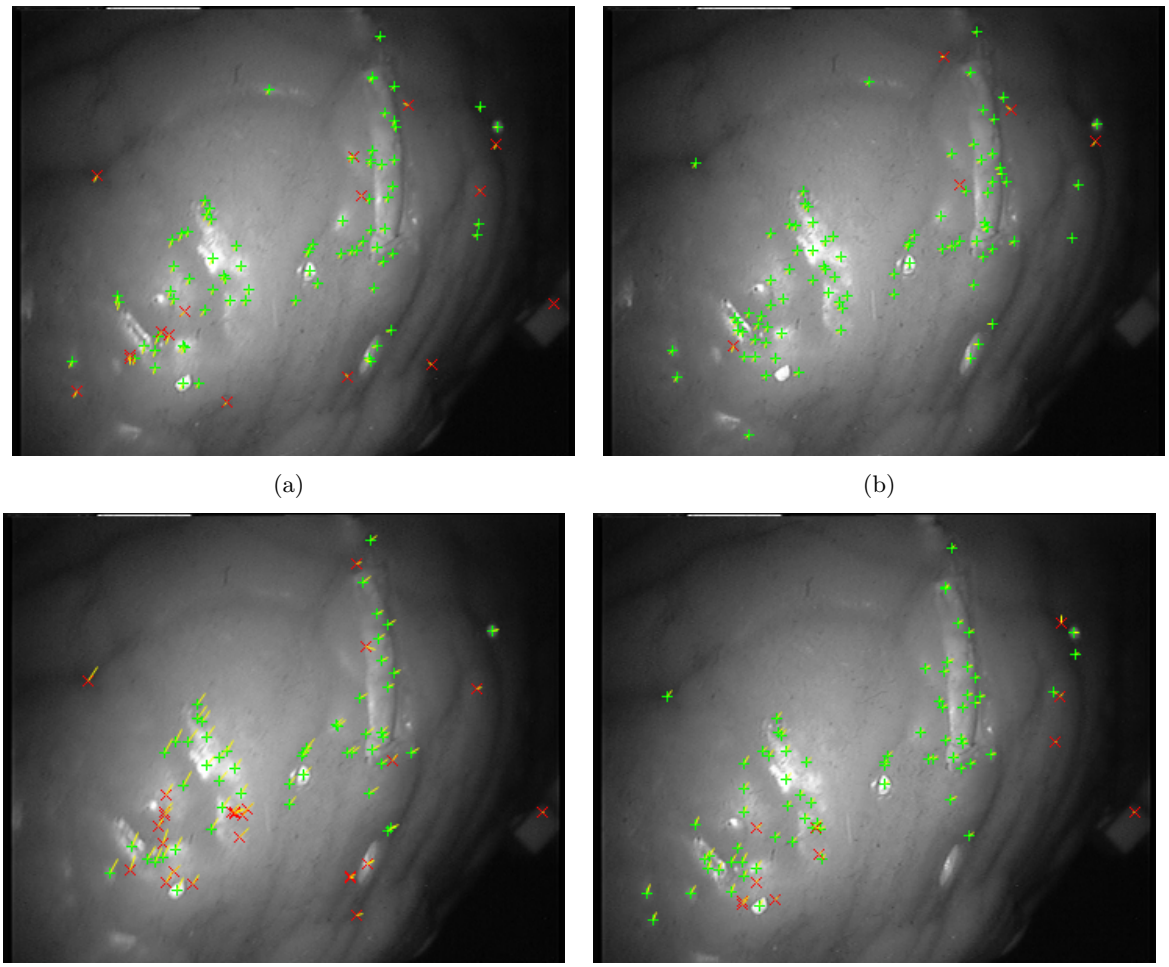


Figure 6.7: Four frames showing detected SURF features and outlier rejection with RANSAC. Red points are outliers while green points are inliers.

We estimate the motion of the heart globally. We use SURF features although we could have used the other features as there is no noticeable difference in regards to the end result. The features are tracked for 300 frames. In figure 6.7 we see four frames with the detected inlier and outlier SURF features, although the surface is not planar most of the points are detected as inliers by the RANSAC algorithm indicating that the motion can be approximated by a linear model. In figure 6.8 we see the sparse optical flow field. The direction of the vectors, mostly those located in the left bottom part, have the most amount of movement while those in back top right have less movement. This is in accordance with the apparent motion.

[1] Source: Hamlyn Centre Laparoscopic / Endoscopic Video Datasets http://hamlyn.doc.ic.ac.uk/vision/



(a) (b)

(c) (d)

Figure 6.8: Four frames 2/25 seconds apart showing the sparse optical flow field. The perceived motion is that of the surface expansions and contractions with a perceived motion of the bottom part moving up and down in the diagonal direction.

Stabilisation

The stabilisation algorithm is applied with a crop window of 90% size of the original frame size. Figure 6.9 shows the estimated original path and the obtained smooth path. We see that the former is in accordance with the expected motion of the periodic cardiac cycle, the latter is a straight line indicating that as a result of the stabilisation we should expect no translation motion in the x and y directions. The estimated original motion seems to indicate that we have a roughly slowly descending trend in both directions.

We test to see the result of the stabilisation with the three different motion models. We find that the best visual stabilisation is obtained with an affine model and that there is a considerable reduction in motion. With the similarity and translation models, there is a reduction in motion however we still observe wobbling. Figure 6.10 shows the mean image of 100 frames for the original video and the stabilised video. We see that the original has blurring due to its motion while the stabilised one has relatively no blurring, indicating that there is an effective reduction of the original motion. In figure 6.11 we see the mean images for 300 frames, we can see the wobbling motion in 6.11 (c) obtained using a similarity model, we also see that this time there is directional blurring in the mean image stabilised using an affine model. However, looking at the inter-frame error in table 6.1, we see that pixel-wise the stabilised videos have a reduction in the mean and a considerable reduction in the variance.

In figure 6.12 we see four frames of the original video with the location of the crop windows showed. The yellow crop window is the 90% size one that we used for the stabilisation algorithm, the red crop window is the maximised one which we have proposed to add so as to prevent cropping more than necessary.

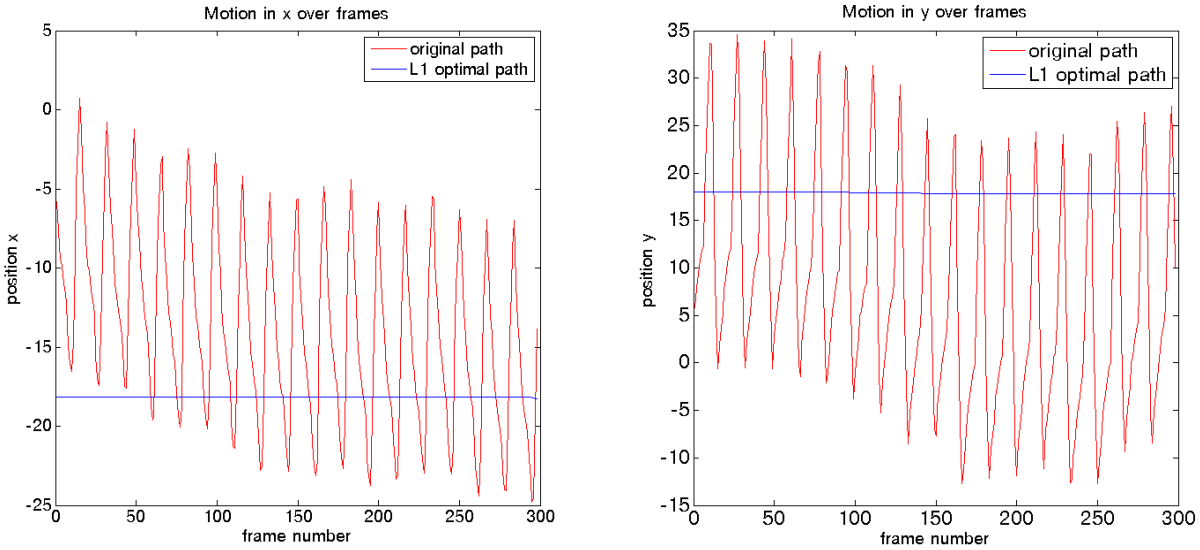
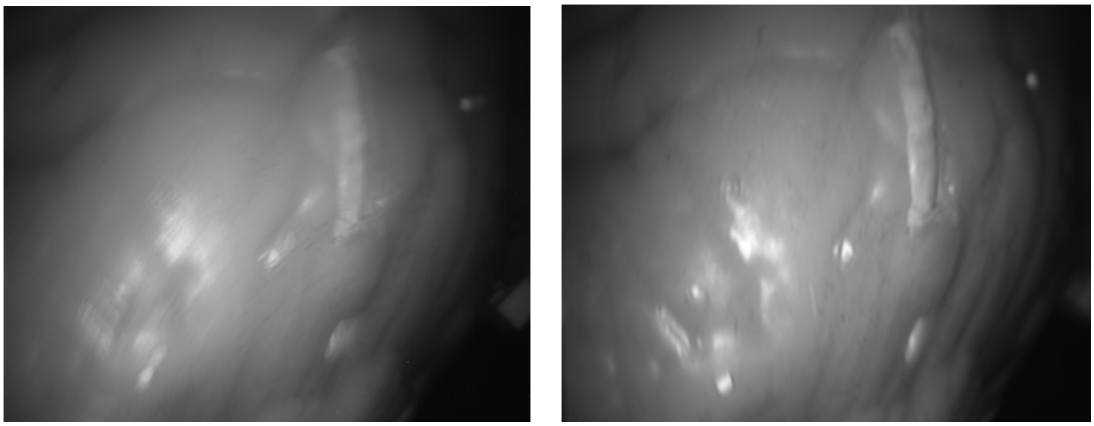
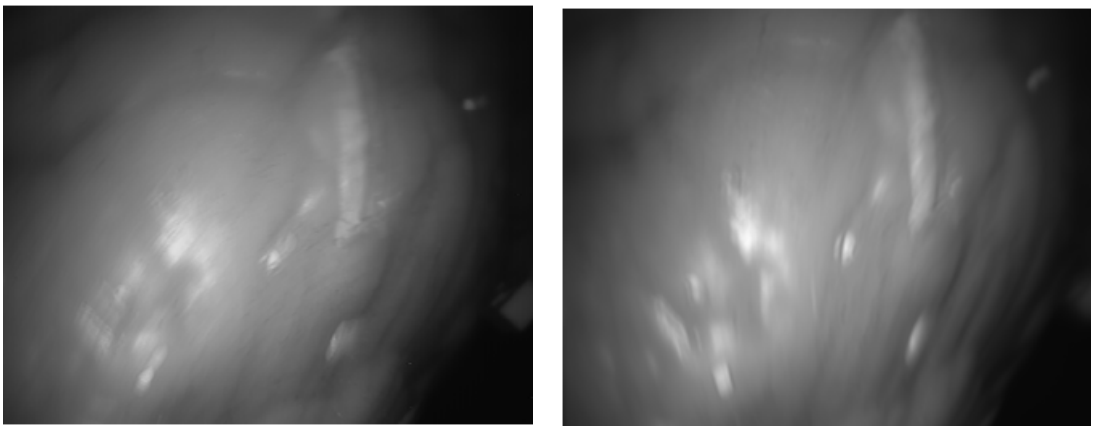


Figure 6.9: Graphs showing the original estimated camera path (in red) and the computed stabilised path (in blue) for the x and y direction over 300 frames using an affine model.



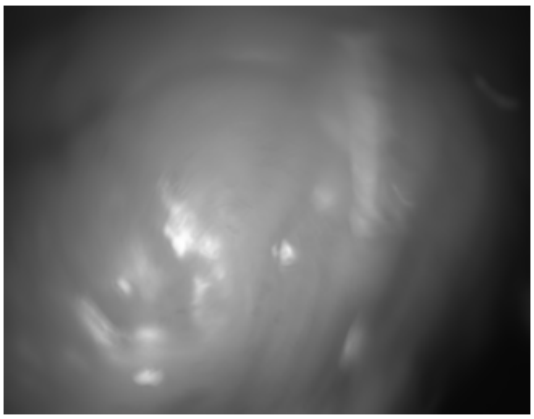
(a) Original video (b) Stabilised video

Figure 6.10: Mean images for 100 frames of the phantom heart video using an affine model



(a) Original video

(b) Stabilised video. Affine model.



(c) Stabilised video. Similarity model.

Figure 6.11: Mean image for 300 frames using an affine model

	100 Frames		300 Frames		
	Original	Stabilised (Affine)	Original	Stabilised (Affine)	Stabilised (Similarity)
Mean	28.7767	5.6208	27.8895	5.0408	5.6365
Variance	302.2192	13.6707	271.9184	9.4420	16.3036

Table 6.1: Inter-frame error for the phantom heart video.

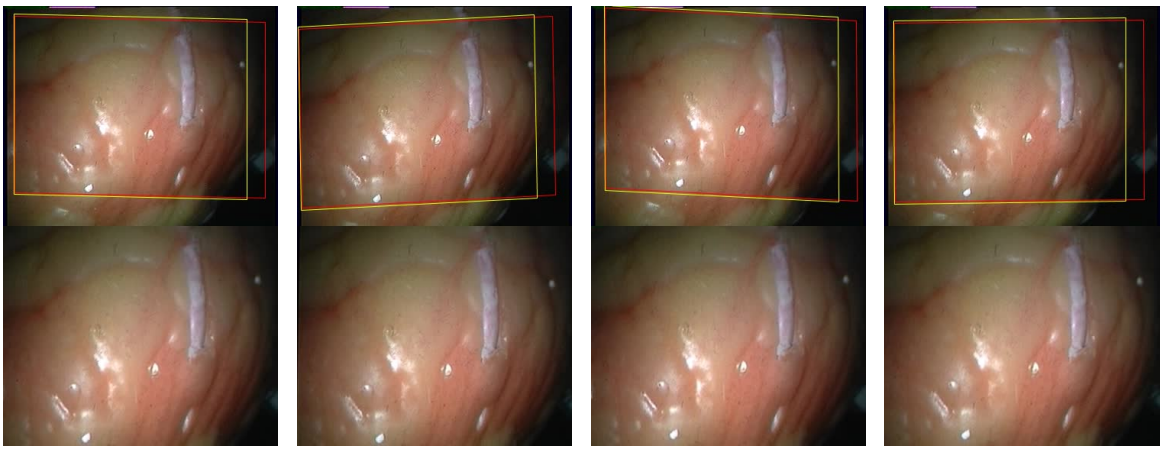


Figure 6.12: Four frames showing the crops window on the original video (above) and the cropped Stabilised video (bellow). The yellow crop window is the 90% size of the original frame used for computing the stabilisation, the red one is the maximized crop window enabling us to avoid cropping more than necessary.

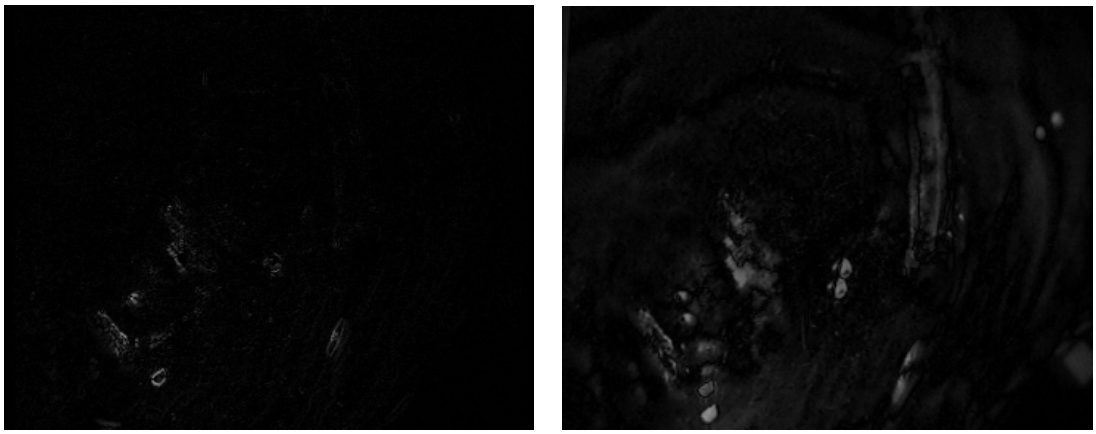
Motion Estimation Error

Looking again at figure 6.11 we see that there is blurring in the mean image of 300 frames using the affine model which was not visible in figure 6.10 with 100 frames. If we look at figure 6.9 we see that the original estimated path has a descending trend as noted previously, however when looking at the original video, it does not have that type of motion. It seems that the motion estimation is not accurate, the optimisation part of the stabilisation algorithm still works as expected as given there seems to be a descent in the x and y directions and that there is room for the crop window to stay fixed, then it can counteract that motion and give a fixed constant path. As a result we see a directional drift in the stabilised video.

The drift is observed for the 3 different motion models (affine, similarity and translation) with the most drift being with an affine model. We have tried increasing the number of detected features on the surface and used the three different feature detectors employed in this project, there was a slight improvement however the drift was not eliminated. This is most likely due to an accumulating error in the global motion estimation and also it being the usual problem with tracking over long periods of time.

To see visually that there is no such movement in the original video, we look at an image difference in figure 6.13 between two frames from the original video and two frames from the stabilised one.

Although the drift might seem small for 300 frames, when we look at the motion estimation for 1000 frames in figure 6.14 we see that over time the effect is more pronounced, with roughly 40 pixels drift in the x direction and 80 pixels drift in the y direction.



(a) Original video (b) Stabilised video

Figure 6.13: Image difference of frame 2 and frame 287 from roughly the same point of the cardiac cycle.

The drift is also present using using the two other motion models. We track an area using the Median Flow tracker for 1000 frames and show the result in figure 6.15 where we notice that there is less drift in comparison to the affine model with roughly 15 pixels in the x direction and 20 pixels in the y direction. In figure 6.16 we show the result of tracking using the TLD tracker, we see that the drift is even less, although the envelope of the path is more jittery due to the corrections to the tracking introduced by the detector.

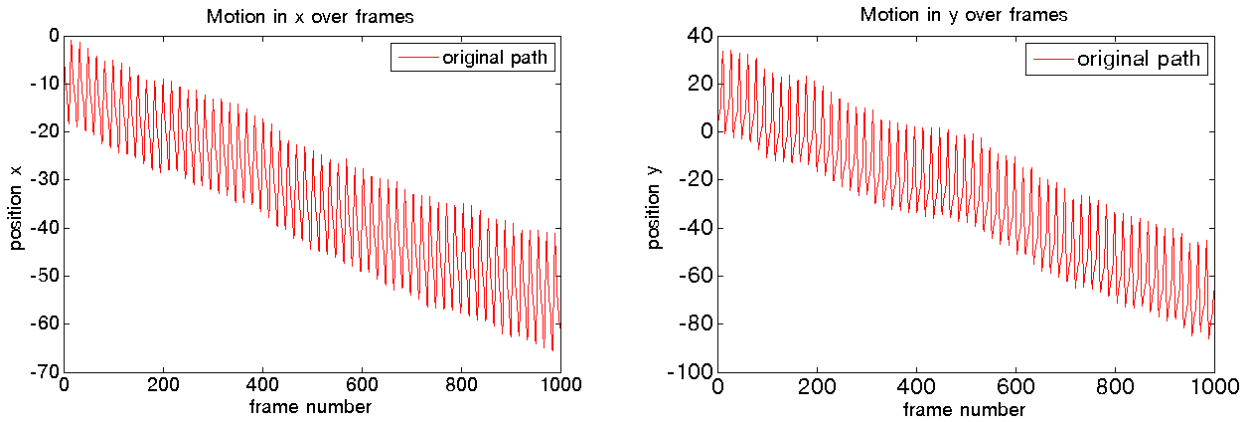
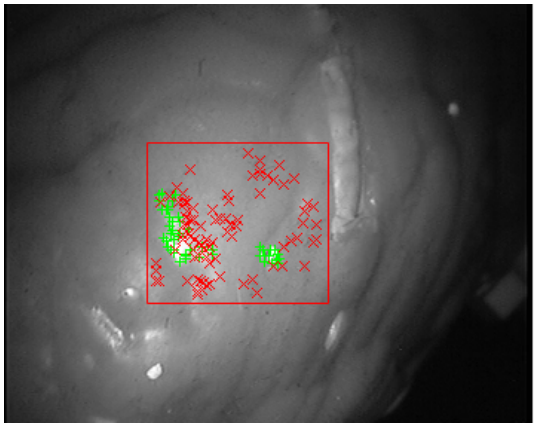
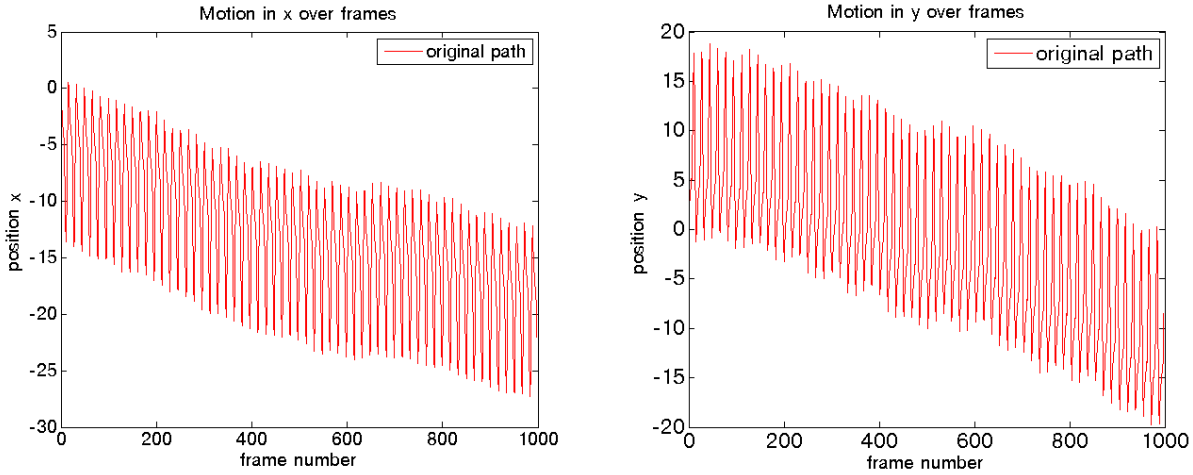


Figure 6.14: Graphs showing the drift in the original estimated camera path for the x and y direction over 1000 frames.



(a) Tracked bounding box. Green points are inliers while the red ones are outliers.



(b) Graphs showing the estimated camera path for the x and y direction.

Figure 6.15: Graphs showing the drift in the original estimated camera path for the x and y direction over 1000 frames. Tracked using Median Flow as an object tracker.

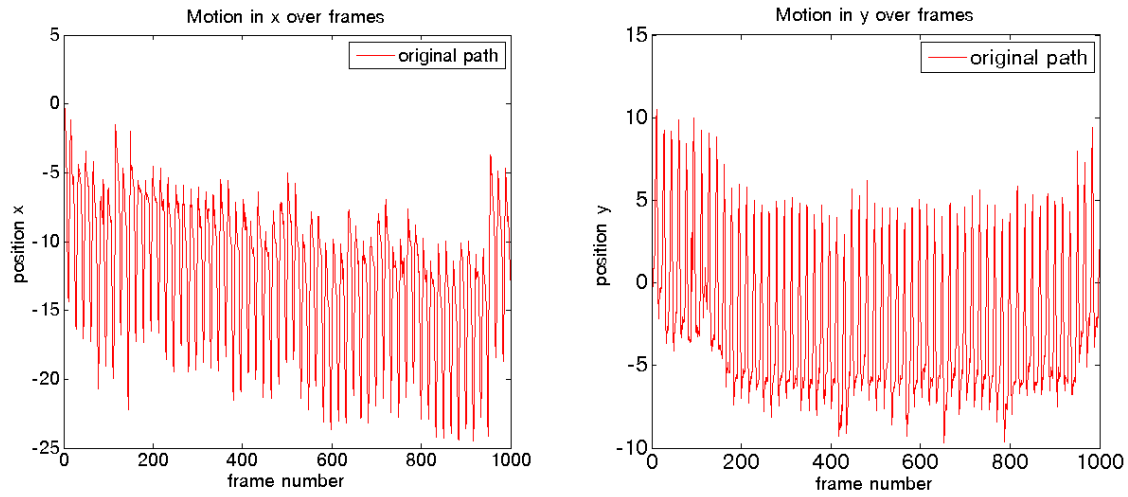


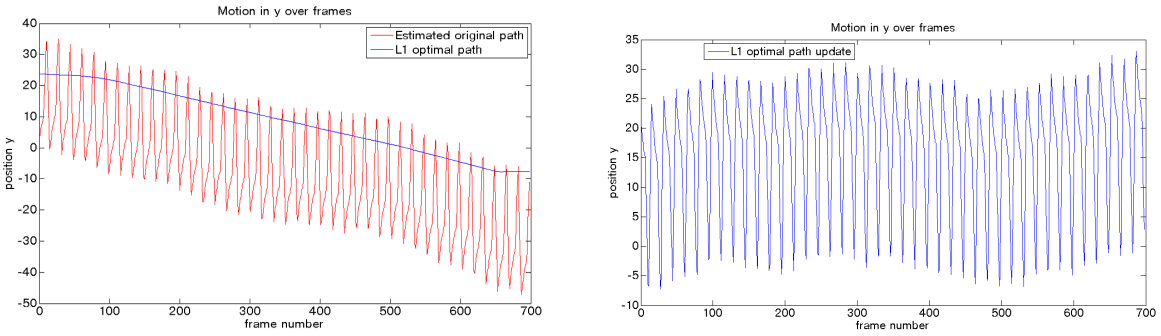
Figure 6.16: Graphs showing the camera path for the x and y direction over 1000 frames tracked using TLD. We see that there is less drift, however the envelope is less smooth, this is due to the correction to the position of the bounding box from the detector.

6.4.2 Evaluation

We have applied the stabilisation algorithm to the phantom heart video and have seen that we can obtain good stabilisation results, with the best visual result being from an affine model. The mean image calculated for 100 frames and the inter-frame error all indicate a reduction in perceived motion.

For longer tracking periods we noticed that there is a drift due an accumulation of errors in the motion estimation which in turn introduces a drift in the stabilised video and ends up becoming noticeable. The larger error observed using an affine model is most likely due to the model having higher DOF, affine models also have a tendency to over-fit [1]. There is an error in the motion estimation to a lesser extent using a translational model. The least error is observed using the TLD tracker which is because the detection part of the TLD algorithm helps to correct the drift from the LK algorithm.

Nonetheless, if the overall motion is captured correctly and even if there is a drift, then it is still possible to have no drift in the end result. The reason why in figure 6.9 we have a resulting optimal path of zero motion in the x direction is that the crop window still had space to move and given that one of the constraints of the stabilisation algorithm is to have constant paths, then the result is a constant path. To see this more clearly we apply the stabilisation on 700 frames using a crop window of 95% to give it less room. We see in 6.17(a) that the smooth path follows along with the estimated drifting path, as a result the update transform that would be applied to the crop window as seen in (b) has relatively no drift.



(a) Graph showing the estimated original path and the smooth path obtained after the update transforms are applied to the estimated original path.

(b) The result of applying the update transforms to the fixed point $(0, 0)$

Figure 6.17: Stabilisation for 700 frames of the phantom heart video with a crop window 95% size of the original.

As a solution, one could possibly add additional saliency constraints such that the crop window follows more in the direction of the estimated path such that even if there is an error in the motion estimation and that the overall motion between consecutive frames is well captured, then it is possible to obtain a good stabilisation result.

Another way to possibly adjust the drift is that we could reapply the stabilisation with a translation model. We tested this and saw it does lead to a considerable reduction in the drift in the case of this video.

6.5 Experiment 3: Real Heart

We now test the performance of the stabilisation algorithm on a video of the real heart.

6.5.1 Set-up and Results

The video

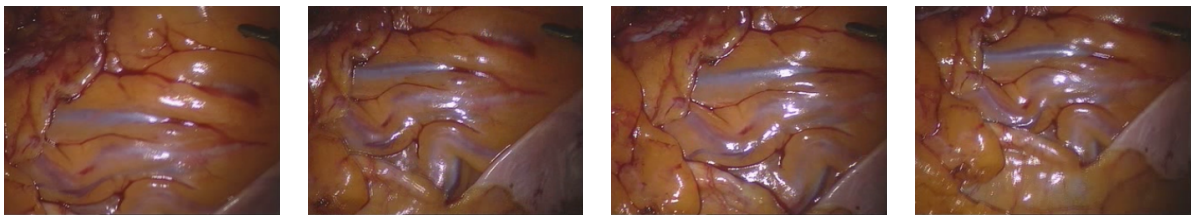


Figure 6.18: Four frames from the real heart video.

The video is of a real heart[2] showing surface deformations and having both cardiac and respiratory motion. The video contains view dependent specular highlights due to the wet nature of the surface. The video size is 300×250 and has a frame rate of 25fps. The video contains a noticeable amount of blurring at certain frames due to the frame rate and the speed of the cardiac contractions. This type of video is an example of the types of sequences that could be encountered in a real setting.

Motion Estimation

We estimated the motion globally. SURF features are used as they are said to be more suited to be tracked with the LK algorithm on the beating heart according to [14]. In figure 6.19 we see the sparse optical flow field, notice the amount of blurring causing the lesser number of detected feature points. Also, during the contractions, we see that there are more points detected as outliers due to the different motion directions of the vectors in different areas.

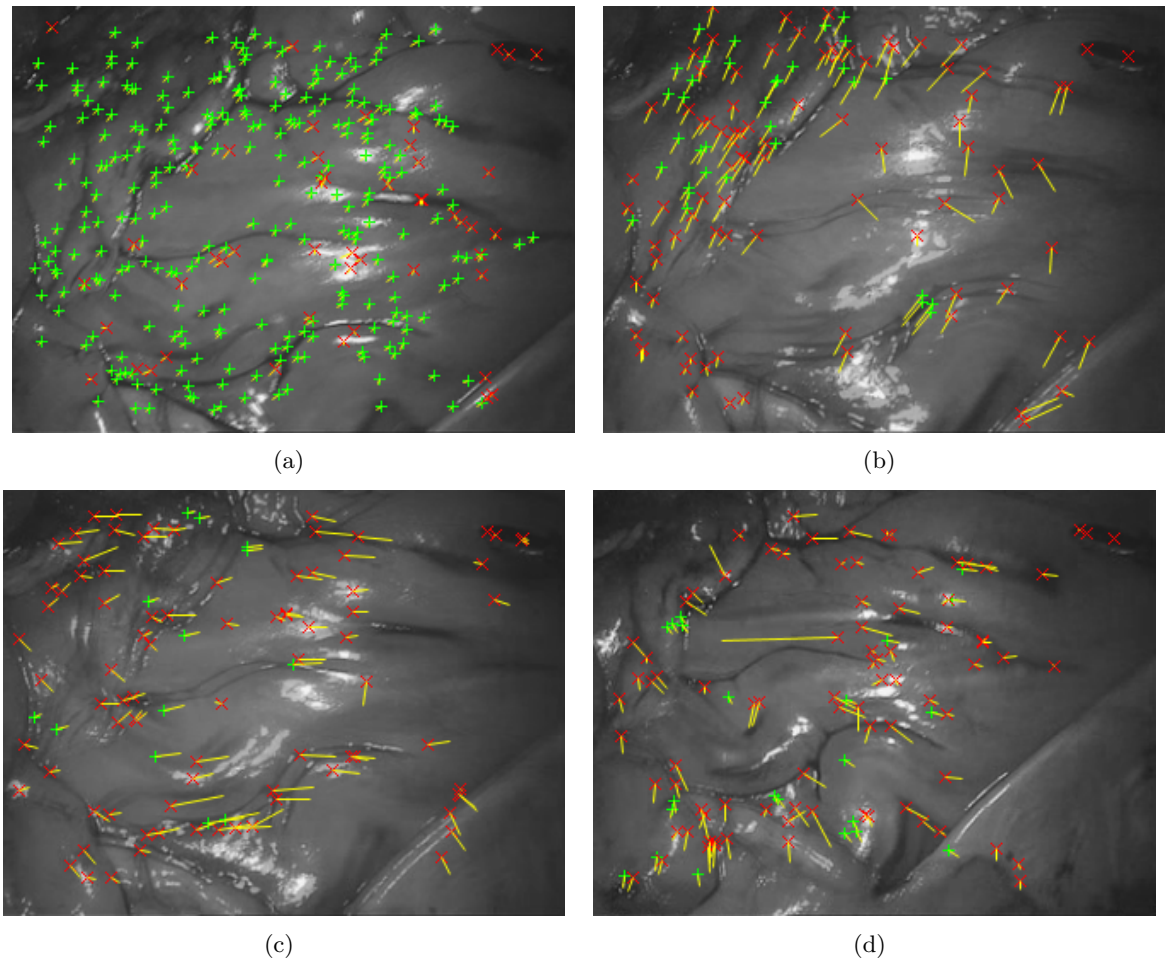
In figure 6.20(b) we see that some features detected in specular highlight areas are considered inliers by RANSAC which means they have an impact on the motion estimation. The impact can be seen more clearly in figure 6.21(a) and 6.21(b) where we show the estimated path obtained from both cases. For the stabilisation, we use the result obtained by excluding the specular areas.

We perform motion estimation using an affine model in one case and a translation model in the other case.

For the affine model, motion estimation is done globally. We found that performing motion estimation using an affine model on a smaller area led to a considerable amount of errors which is mostly due to the lower feature point count, attempting to increase the detected points did not lead to an improvement as already, in order to obtain a sufficient number of points, the feature detector was set with a really small threshold value.

With the translation model, we estimate it as per the Median Flow tracker, whereby the translation is calculated as the median of the displacement of the inlier points. We found that performing tracking globally or selecting a smaller area (around 50% of the original frame size) lead to somewhat similar results, but on a smaller scale, we get slightly different results due to each part of the surface moving a bit differently.

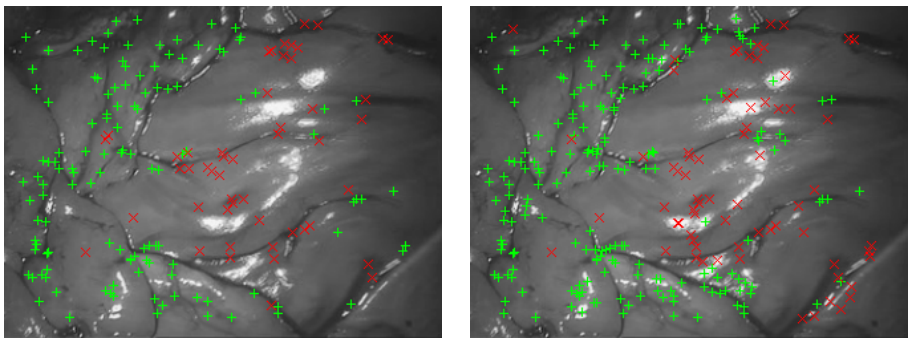
[2]Source: Dr. Eddie Edwards, St. Mary's Hospital



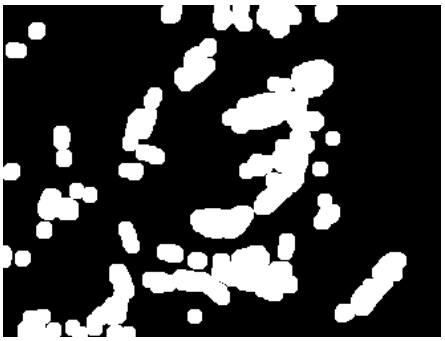
(a) (b)

(c) (d)

Figure 6.19: Four frames 2/25 seconds apart showing the sparse optical flow field.

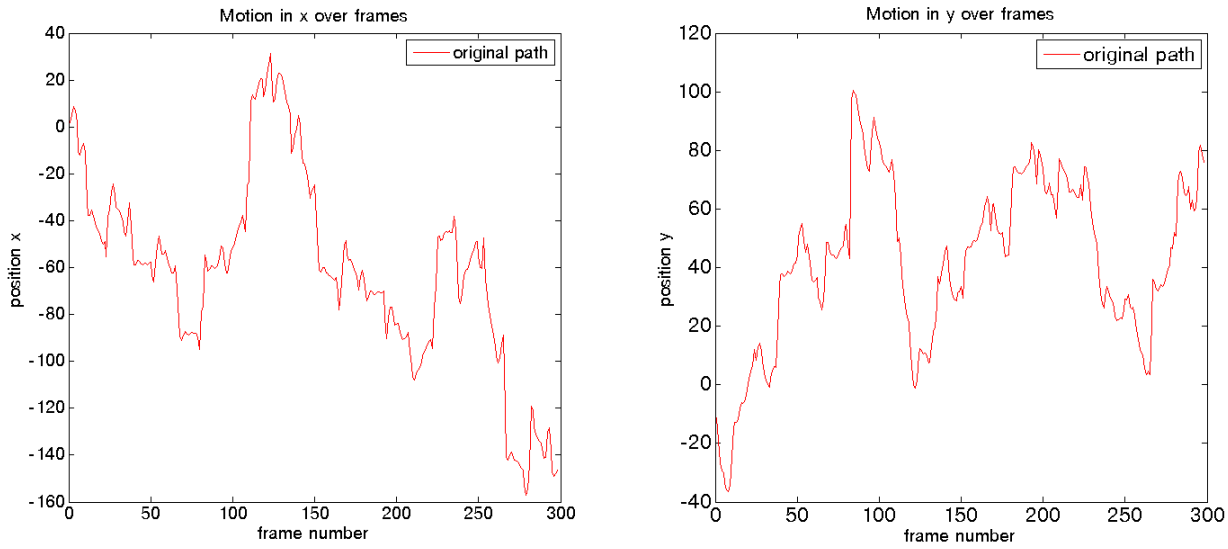


(a) With specular highlight detection (b) Without specular highlight detection

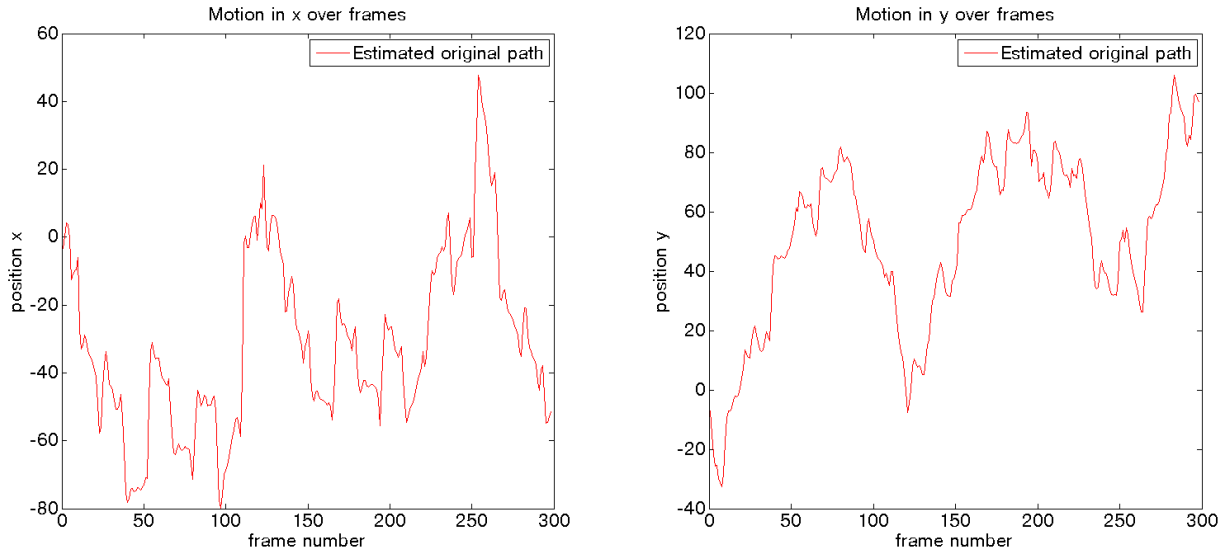


(c) Mask of the specular highlight areas

Figure 6.20: Images showing the detected features on the heart video. Green points are inliers, red points are outliers.



(a) Estimated global motion in the x and y direction without specular removal.



(b) Estimated global motion in the x and y direction obtained by excluding the specular areas from the tracking.

Figure 6.21: Effects of specular highlights on the motion estimation (Affine model).

Stabilisation

We apply the stabilisation using a crop window of 70% size of the original video. The resulting obtained smooth path for an affine model can be seen in figure 6.22 and for a translation model in figure 6.23. We see that with an affine model the obtained smooth path does not result in a zero motion while with the translation model we have an almost zero motion. The reason why with the affine model we still have motion in the stabilised path is that at some points the crop window reaches a boundary so it would have to move.

In figure 6.25 we see the mean images obtained for the original video and the stabilised one. In-painting was used to get a clearer image as otherwise specular highlight would have introduced white blotches. In (a) and (b), for 300 frames it is almost all a blur, although we can make out traces of lines in the stabilised one. In order to get a better visual, we adjust the contrast of the images by remapping the intensities such that 1% of the pixels are saturated at low and high intensities. We see more clearly in (c) and (d), where both images exhibit blur, but it is a bit less with the stabilised one. An important thing to note is that, unlike the phantom heart which had no surface deformations, the tissue deformation will introduce blurring in the mean images due to the contractions even if the resulting video is well stabilised. So we also look at the mean for 80 frames in (e) and (f) which should have lesser amount of blurring from the contraction, as a result we can see more distinct areas indicating that more images are aligned, notably in the top left part.

Figure 6.26 shows the mean images obtained using the translation model.

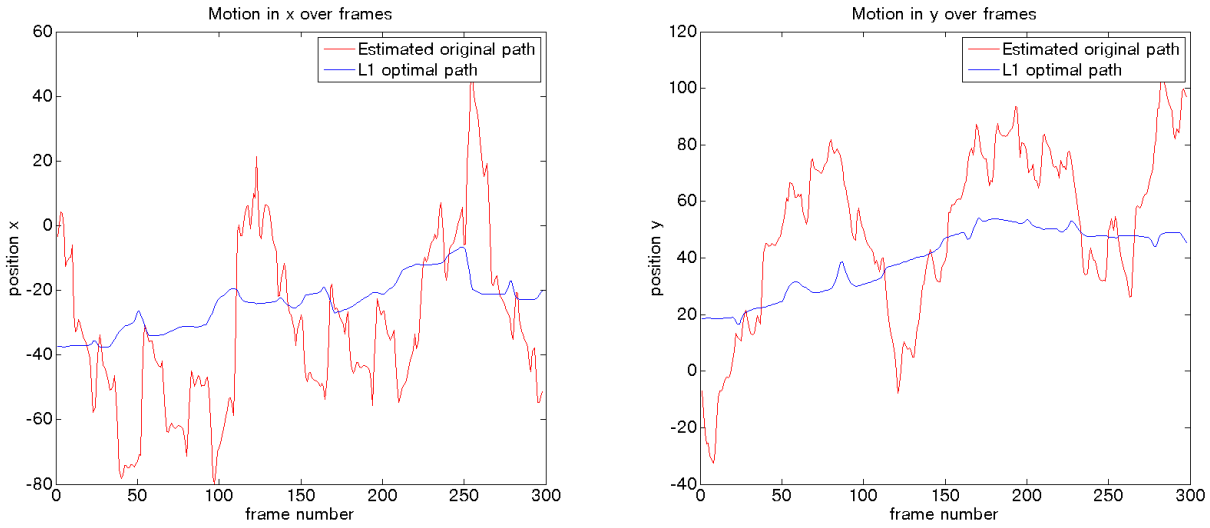


Figure 6.22: Graphs showing the estimated original path and the corresponding smooth path in the x and y direction.

Table 6.2 shows the inter-frame error, we see that there is reduction in both the variance and the mean indicating that there is less inter-frame motion, we used in-painting as otherwise the high values of the specular would have falsely increased the amount of error.

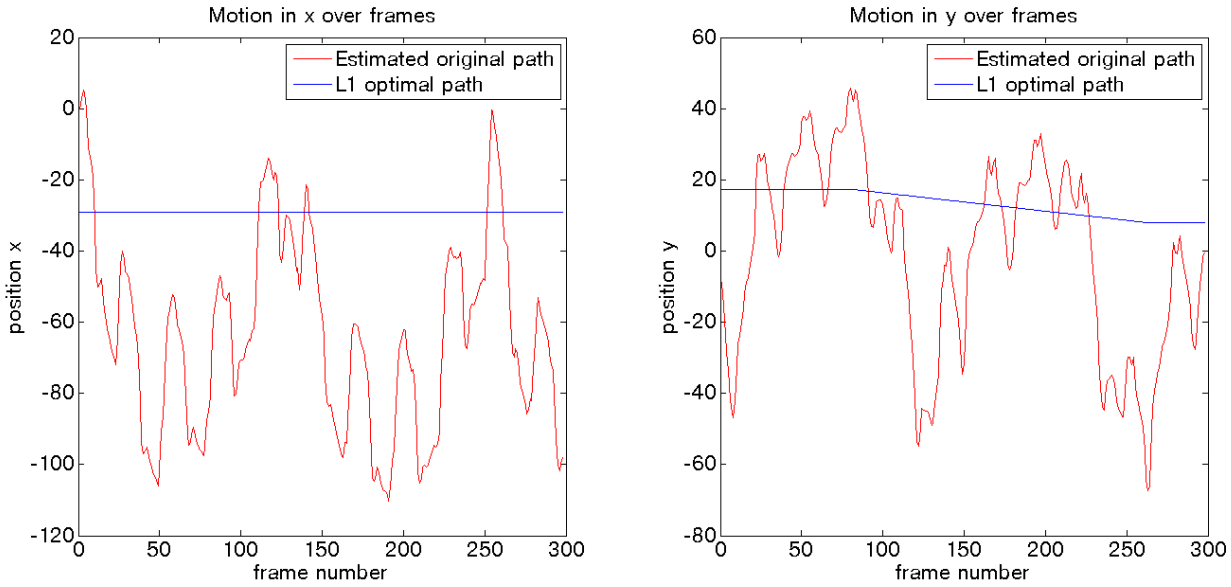


Figure 6.23: Graph showing the estimated and smooth paths in the x and y direction using a translation model.

	300 Frames			80 Frames		
	Original	Stabilised (Affine)	Stabilised (Translation)	Original	Stabilised (Affine)	Stabilised (Translation)
Mean	84.3809	24.5556	19.8255	67.4547	23.1436	19.2247
Variance	1260.2893	353.7194	250.633681	2429.8824	633.6314	289.1361

Table 6.2: Inter-frame error for the real heart video.

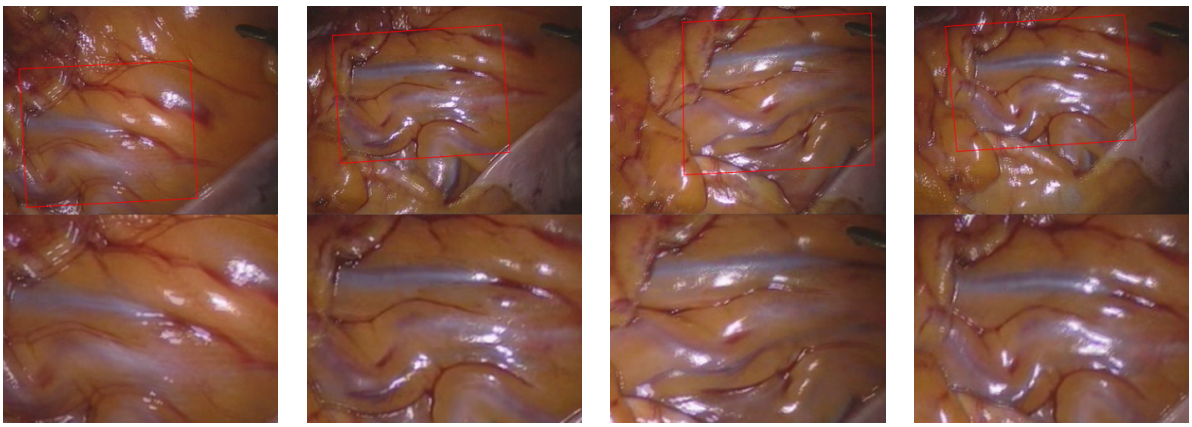


Figure 6.24: Four images showing the crop window on the original frames with the corresponding stabilised frames.

(a) Original video - 300 frames

(b) Stabilised video - 300 frames

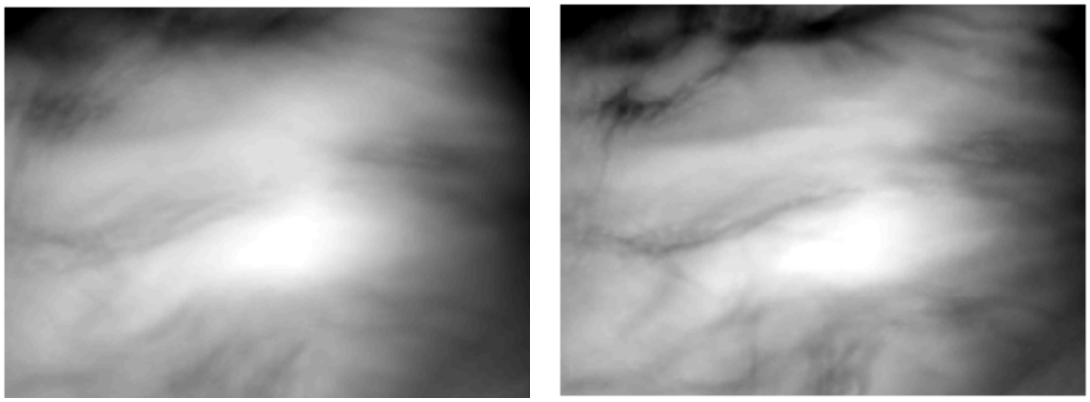
(c) Original video - 300 frames (with contrast adjustment)

(d) Stabilised video - 300 frames (with contrast adjustment)

(e) Original video - 80 frames (with contrast adjustment)

(f) Stabilised video - 80 frames (with contrast adjustment)

Figure 6.25: Mean images for 300 frames using an affine model. In-painting was used.



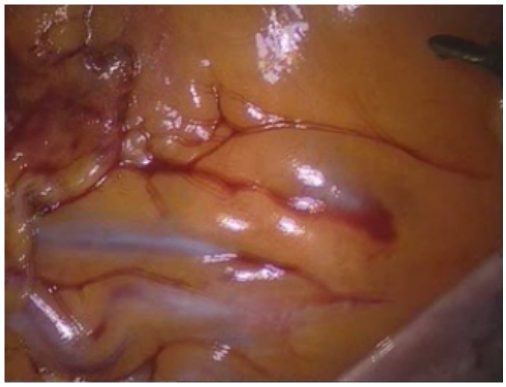
(a) Stabilised video - 300 frames

(b) Stabilised video - 80 frames

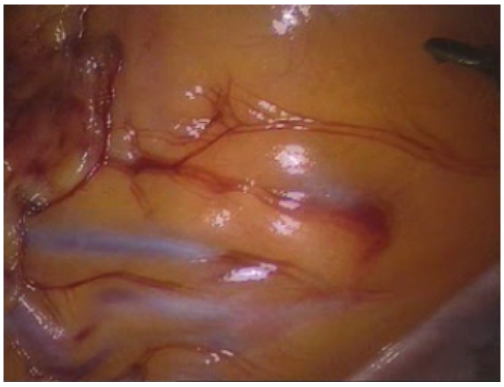
Figure 6.26: Mean images for 300 frames and 80 frames using a translation model. In-painting was used.

Motion Estimation Error

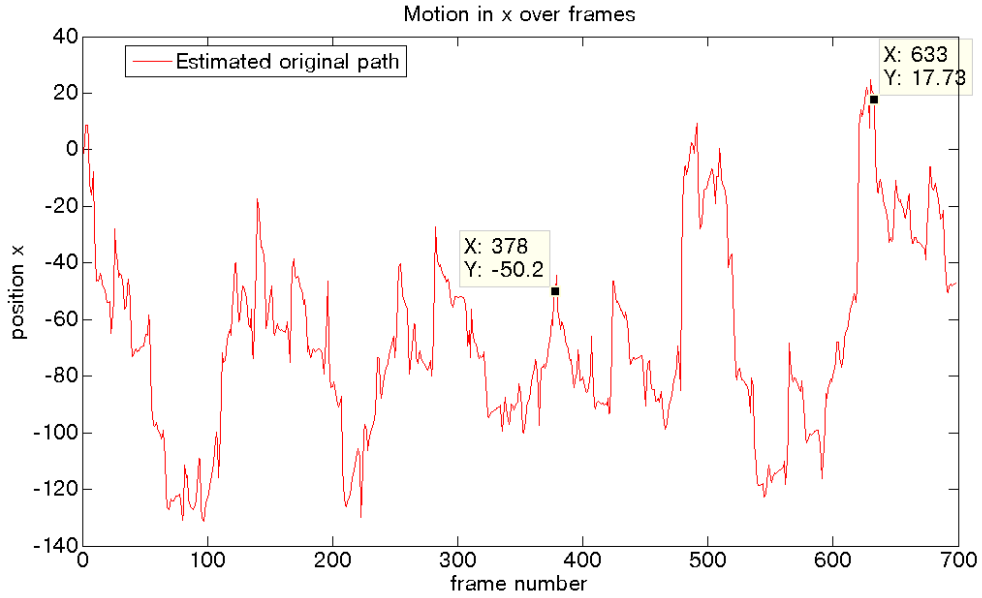
Besides the issues from specular highlights, the motion estimation tends to accumulate errors. We can see that more clearly by tracking for a longer period. In figure 6.27 we see the result of motion estimation in the x and y directions both for the affine and translation models. The position of two images taken from the video that should have the same position are not so in the estimated path.



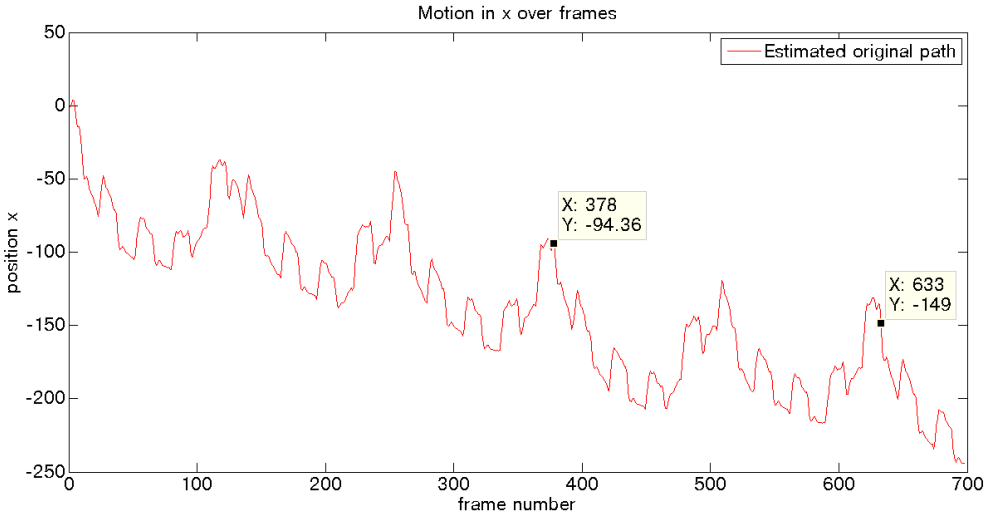
(a) frame 378



(b) frame 633



(c) Estimated path in the x direction using an affine model. The position of the images is indicated on the graph.



(d) Estimated path in the x direction using a translation model. The position of the images is indicated on the graph.

Figure 6.27: Tracking carried out on 700 frames with the specular areas excluded. We see that although the images should be at about the same position we have an error in the estimated position.

6.5.2 Evaluation

We saw here that the results for stabilisation obtained for the real heart are not as good as those we had obtained for the phantom heart due to the more complicated nature of the surface deformation, nonetheless there is a noticeable reduction in cardiac and respiratory motion as seen in the mean images and the inter-frame error, with the translation model giving the least inter-frame error.

Motion estimating is an important step for the stabilisation. We have seen that specular highlights have a considerable effect on the global motion estimation and eliminating them from tracking does lead to an improvement.

When it comes to capturing the overall motion, an affine model performs worse over time. This is most likely due to the higher DOF which introduces a considerable amount of error over longer periods. The result of this is that we end up with residual motion in the final video which manifests more and more over time. Also another reason to consider is that since the surface deformation is non-linear, capturing the motion globally with a just an affine model is not well suited. Another type of non-linear motion model that could capture the deformation more accurately might be needed to improve the stabilisation.

Using a translation model for the stabilisation gives a better result in capturing the overall motion and eliminating somewhat most of the respiratory motion and some of the cardiac motion. There remains a small residual motion in the final video but it is lesser than with using an affine model. However with the translation model, we have no compensation for scale or rotation.

We should also note that the quality of the video is not good whereby there is a considerable amount of motion blur due to the speed of the contractions which makes tracking more difficult. Using a better quality video could possibly lead to an improvement in the motion estimation.

6.6 Conclusion

An important part that determines the effectiveness of the stabilisation is the motion estimation. We have seen that for a video containing a rigid background, such as that in experiment 1, we can acquire a good estimate of the global motion using a linear model and that the outlier rejection scheme is able to eliminate points not belonging to the background. The end result of the stabilisation is a video which is smooth and jitter free.

In the case of the phantom heart we can get a global motion estimate which describes well the movement of the beating synthetic heart, there is however an accumulation of error in the motion estimation leading to a slow drift in the final video. The result depends on the dimensions of the original crop window. We saw that having a smaller crop window allows the smooth path to follow along with the drift resulting in no almost no drift in the final video.

When applied to the real heart we see that we had to deal with more issues that affect the motion estimation, notably the specular highlights as well as the blurring caused by the speed of the periodic contractions which is not possible to capture clearly with just 25fps. Nonetheless, despite the accumulating errors in the motion estimation, we can see a noticeable reduction in motion. A translation motion model has a lesser amount of deteriorating motion estimate over time due to the lower DOF than it is the case with an affine model, at the cost of not compensating for scale and rotation, but it still does capture better the overall motion.

An important thing to note that, in the real setting, other types of situation could also be encountered that would introduce more errors in the global motion estimation, such as the introduction of instruments which would inevitably be present and other related issues as mentioned in section 2.2.1.

Chapter 7

Results and Evaluation II

In this chapter we perform a second set of experiments. Experiment 4 is aimed at investigating predictive stabilisation using K-NN based on the L1 optimal paths stabilisation algorithm.

In experiment 5 we investigate the effectiveness of obtaining a motion estimate using manifold learning and using it to perform post and predictive stabilisation.

In experiment 6, we test the effectiveness of the TLD tracker at offering real-time stabilisation for small tracked areas.

7.1 Experiment 4: Predictive Stabilisation Using K-NN

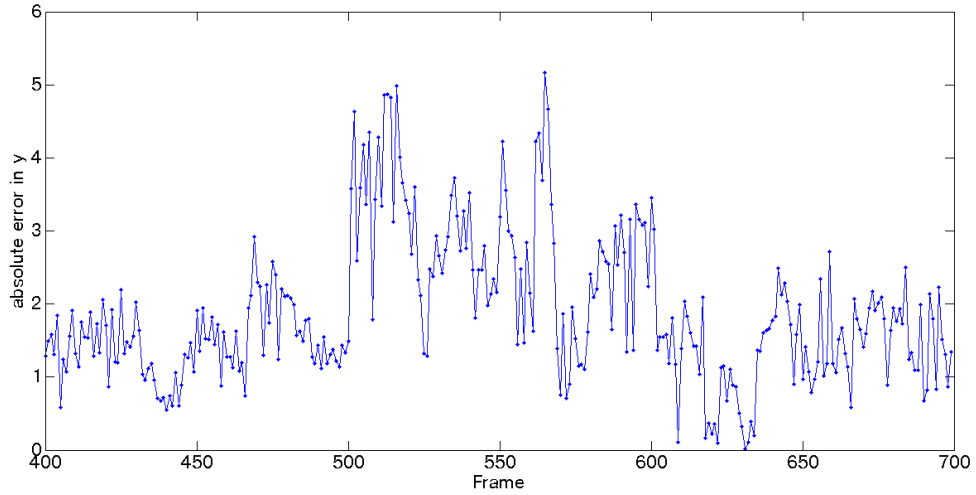
In this experiment we test the effectiveness of applying predictive stabilisation using K-NN as described in section 4.6.1.

7.1.1 Phantom Heart

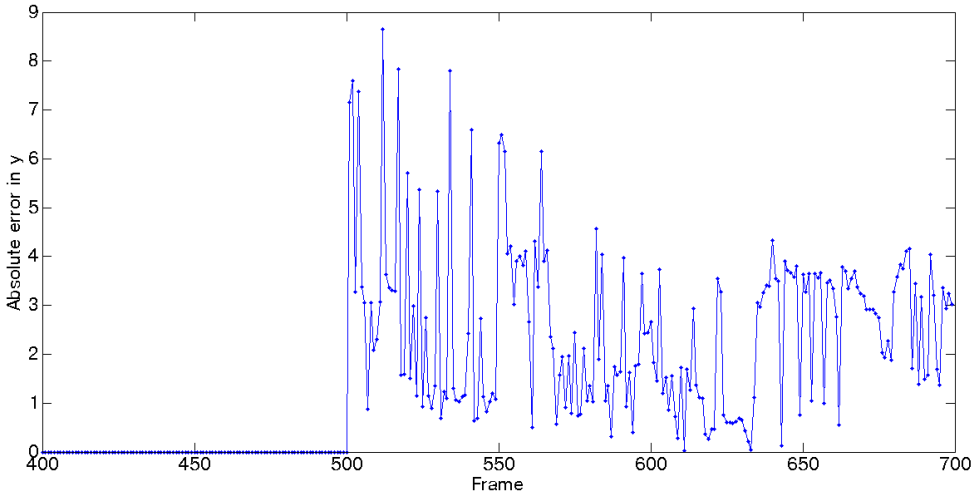
We use the first 500 frames for training and 200 frames for testing. The result is compared to the stabilisation of all the 700 frames. For the stabilisation we use a crop window 95% size of the original. Figure 7.1 shows the absolute error for $K = 5$ and $K = 1$. We see that there is a bit less error with the former than the latter, however there a small amount of error with $K = 5$ on the training set (frames 400 to 500). The error manifests in the video with small constant jitter. Table 7.1 shows the mean and variance of the absolute error in the x and y direction.

	x direction	y direction
Mean	2.7170	1.8813
Variance	1.9693	0.9761

Table 7.1: Absolute error in position for the phantom heart video using K-NN for predictive stabilisation. (K = 5)



(a) K = 5.



(b) K = 1.

Figure 7.1: Absolute error in the y direction for the phantom heart video using K-NN for predictive stabilisation. Frames 400 to 500 belong to the training set. Frames 500 to 700 belong to the test set.

7.1.2 Real Heart

We perform L1 optimal paths stabilisation on 700 frames using a crop window 70% size of the original. We then use 500 frames for training and the last 200 frames for testing. We then compare the result of the stabilisation with the original stabilisation. We take $K = 7$ as it gave the least amount of error. Figure 7.2 shows the absolute error in the y direction. We see that from frame 400 to 500 the error is relatively zero, which is expected as they belong to the training set. Starting from frame 500 we see that there is a non negligible amount of error and the curve is not smooth, this manifests itself as constant jitter in the final video. Table 7.2 shows the mean and variance of the absolute error in the x and y direction. The effect is worse if the specular highlights are not excluded.

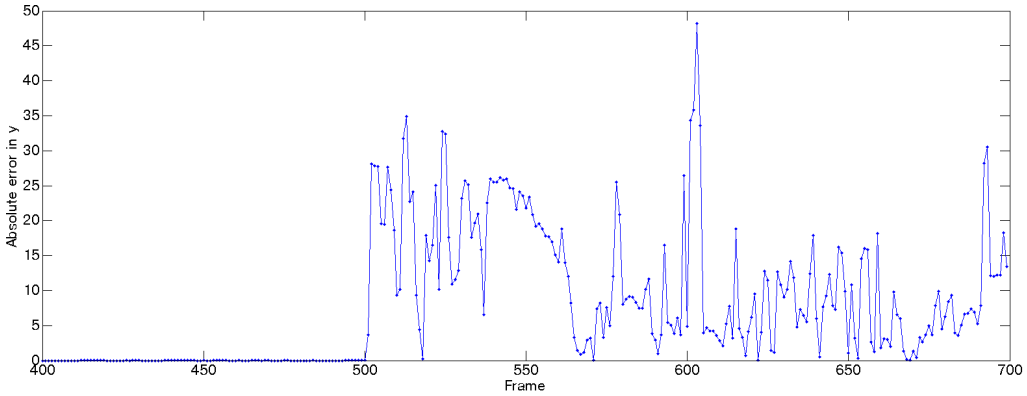


Figure 7.2: Absolute error in the y direction for the real heart video using K-NN as predictive stabilisation based on closest image match.

	x direction	y direction
Mean	8.70	7.95
Variance	119.30	90.01

Table 7.2: Absolute error in position for the real heart video using K-NN for predictive stabilisation.

7.1.3 Evaluation

We have seen in this experiment that we are not able to obtain good stabilisation results using the nearest neighbour approach.

Even though there is small absolute error in the case of the phantom heart, the obtained video still exhibits jitter. The main reason why we do not get a good result with the phantom heart is because of the error in the motion estimation. To test for this, we redid the experiment using a corrected motion estimate, the error obtained was really small and the jitter was almost eliminated.

In the case of the real heart, the technique performs even worse. Perhaps we could have obtained a better result had we used more frames for training but we were limited by the length of the video.

This was the simplest idea by which we could attempt to do predictive stabilisation and it turned out that it does not work. Other methods could be more suited. For instance we could reduce the amount of jitter by making the amount of stabilisation applied to the current frame depend on the value applied to the previous so as to prevent having erratic motion. An improvement could possibly be obtained if instead of looking for the closest image matches for one frame, we could look for the closest matched sequences of images. However we did not test them in this project. Other approaches are required.

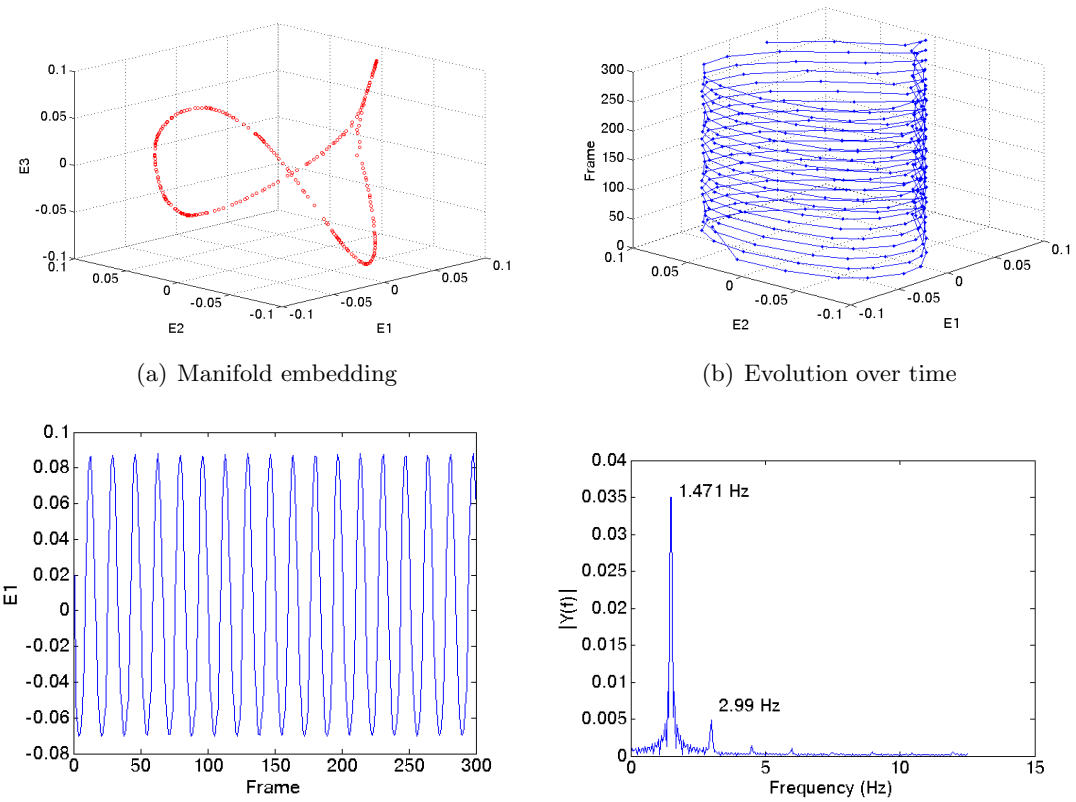
Also, a limitation of this technique is that it is computationally expensive because we are looking for the nearest neighbour using the euclidean distance between each of the images. Rescaling the images to a smaller size does help but it could lead to less accurate results. Instead of comparing entire images, another method would have been to perform tracking to obtain the position and based on the position apply the corresponding update transform, however when we tried this, we obtained worse results due to the errors in the motion estimate.

7.2 Experiment 5: Using Manifold Learning

In this experiment we explore whether it is possible to obtain a good overall motion estimate using manifold learning. We also test the effectiveness of the technique to perform post and predictive stabilisation.

7.2.1 Phantom Heart

We perform dimensionality reduction for 300 frames using Laplacian Eigenmaps with 3 dimensions and 8 nearest neighbours.



(a) Manifold embedding

(b) Evolution over time

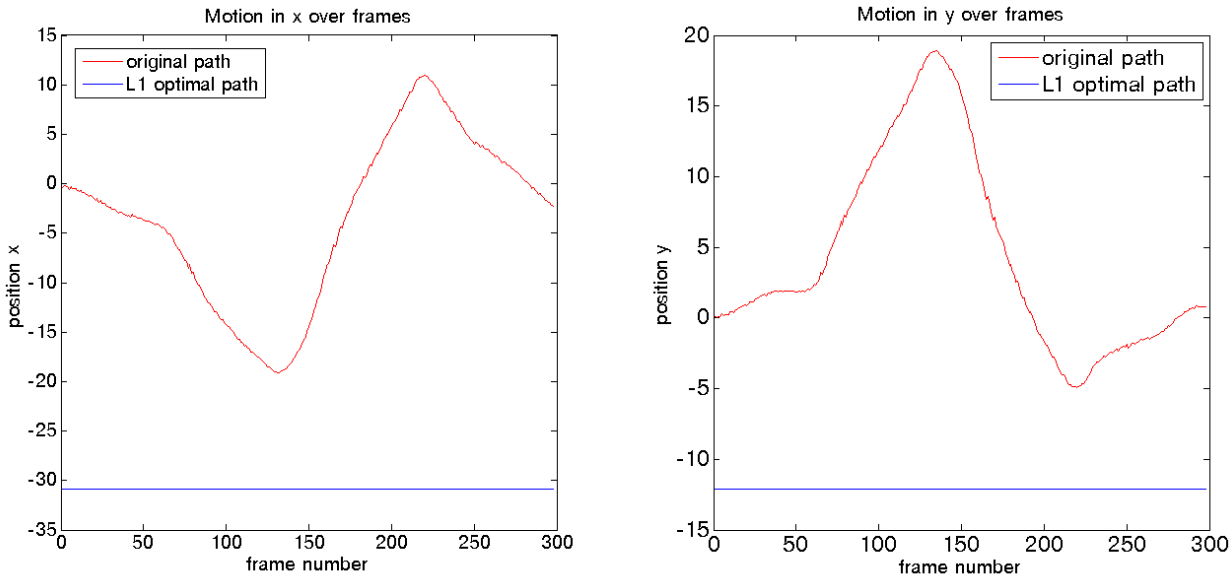
(c) Evolution over time for the first component $E1$

(d) Amplitude Spectrum of the first component $E1$

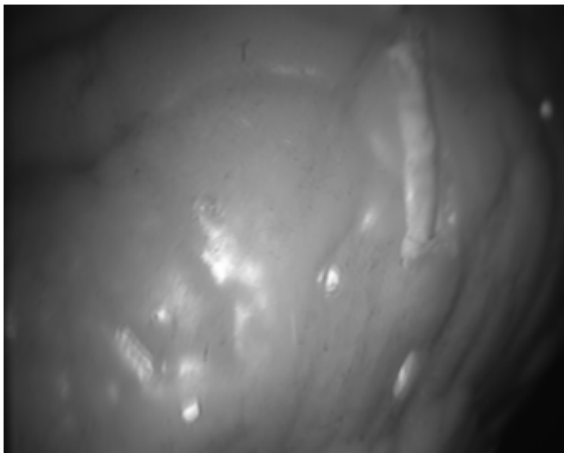
Figure 7.3: Manifold obtained using 300 frames of the phantom heart video.

Motion Estimation Using Image Reorganisation

Looking at the obtained manifold in 7.3(a) we see that the points lie on a closed path. One interesting thing we could do is reorganising the images according to their order along the manifold. Doing this, we obtain a simulated high frame rate video of one cardiac cycle. When estimating the camera path for the reorganised images, we see in figure 7.4 that the drift from frame 1 to frame 300 is roughly 3 pixels in the x direction and 1 pixel in the y direction.
The result is interesting in the sense that we can now simply apply the stabilisation and reorganise the frames thus effectively having an extremely small error in the motion estimation. However this only works well here because we are dealing with an ideal case such as that with the phantom heart which has exactly the same repetitive motion for the entire video. By reorganising the frames, the inter-frame displacement was effectively reduced, making the optical flow estimation more accurate.



(a) The estimated original path and the stabilised path in the x and y direction for the reorganised video.



(b) Mean image for 300 frames using an affine model. There is almost no distortion.

Figure 7.4: Image reorganisation according to their order on the manifold.

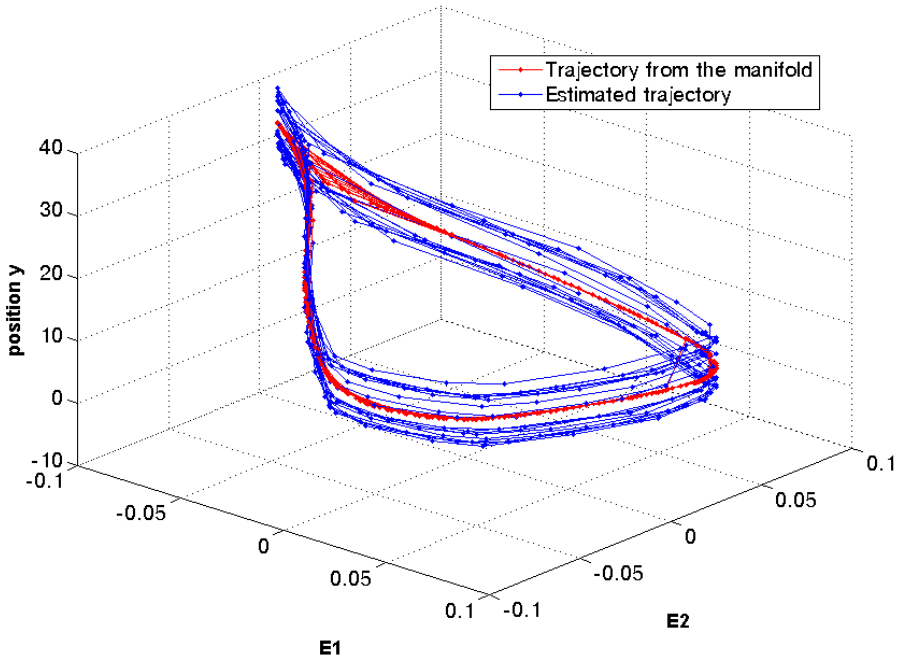
	300 Frame		
	Original	Stabilised (according to manifold)	Stabilised (back in original order)
Mean	27.8895	2.2598	5.6733
Variance	271.9184	0.9902	15.9995

Table 7.3: Inter-Frame Error after reorganising the images of the phantom heart.

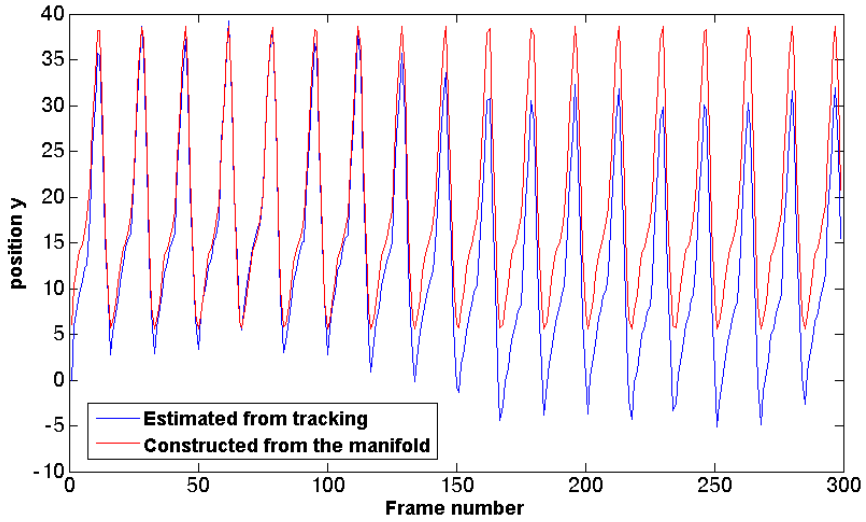
Trajectory Reconstruction

Global We estimate the translation motion globally in the original video by applying the Median Flow tracker on 300 frames. We then perform least squares regression to obtain an expression for the x and y coordinates as a linear function of the embedded coordinates.

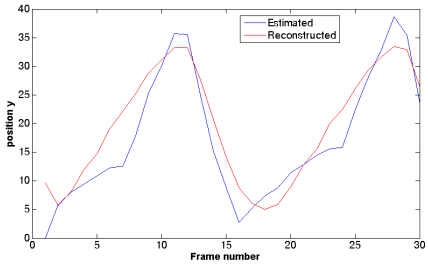
Figure 7.5 (a) shows the original estimated position y as a function of the embedded coordinates $E1$ and $E2$ in blue, the result obtained by performing regression is in red. Notice how the blue trajectory seems to be following as per the shape of the manifold but it is as if it is drifting which is expected because of the drift in the tracking. The obtained red trajectory has no drift. Figure 7.5 (b) shows the position y as a function of the frame number for the original estimated tracked position (blue) and the one obtained as a function of the embedded coordinates (red). We see that the obtained shapes are similar. Figure 7.5 (c) shows the effect of the number of embedded coordinates on the shape of the obtained path. We see that with 3 components we manage to get a similar shape.



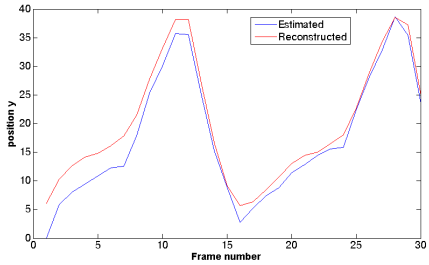
(a) Graph showing the position y as a function of the first two components E1 and E2. The estimated trajectory from tracking of 300 frames is in blue. The red trajectory is the one obtained with regression.



(b) Graph showing the estimated motion (in blue) and the reconstructed motion using the first 3 components from the manifold (in red). Notice how there is no drift in the reconstructed position.



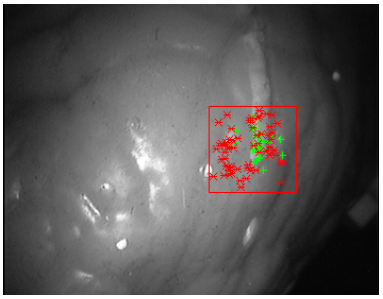
(c) Reconstructed position using two components



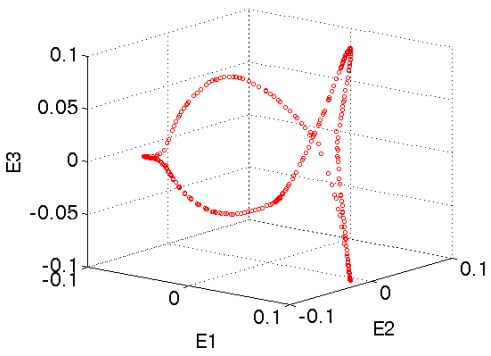
(d) Reconstructed position using three components

Figure 7.5: Trajectory reconstruction using the manifold.

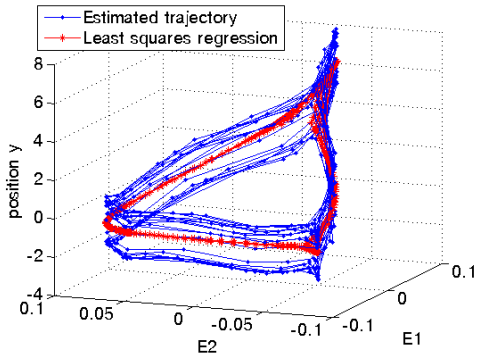
Local We now perform tracking on a local region on 300 frames of the video. We also crop that area and use the images to construct the manifold. We see that the manifold obtained has a slightly different shape than the one obtained globally. We see the results in figure 7.6.



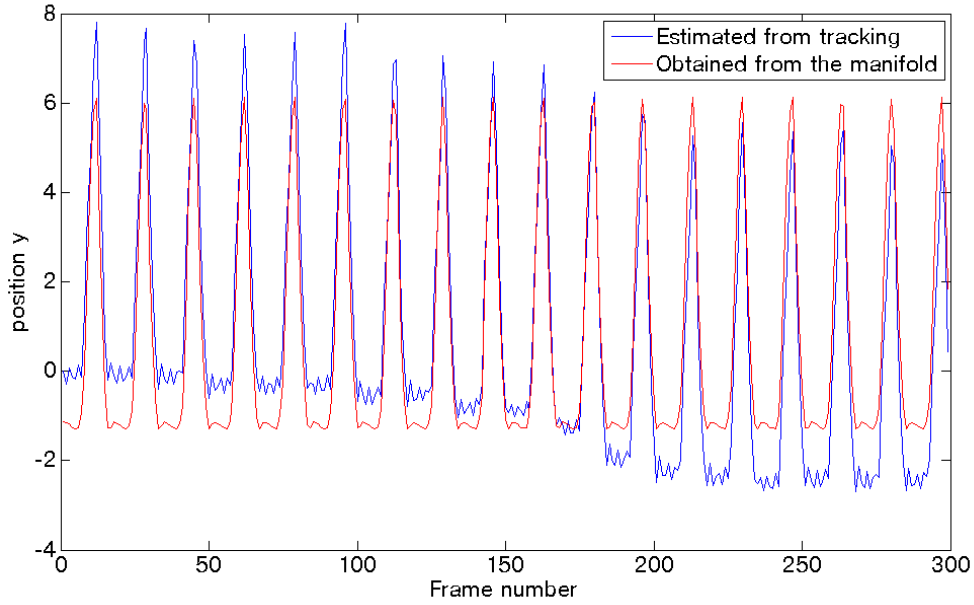
(a) Tracked area. The bounding box is fixed.



(b) Obtained manifold for the the region shown in (a).



(c) Graph showing the position y as a function of the first two components E1 and E2 from the manifold. The estimated trajectory from tracking of 300 frames is in blue. The red trajectory is the one obtained with regression.

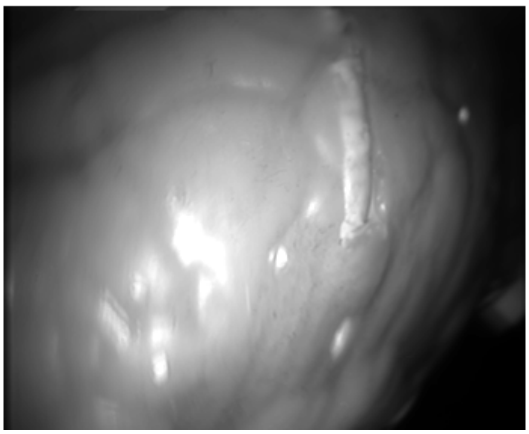


(d) Graph showing the position y as a function of the first two components E1 and E2 from the manifold. The estimated trajectory from tracking of 300 frames is in blue. The red one is obtained as a function of the embedded coordinates.

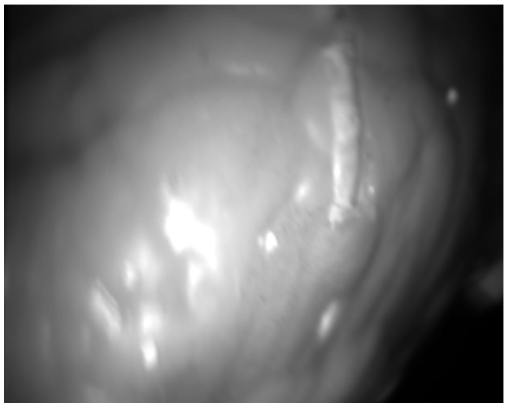
Figure 7.6: Trajectory reconstruction using manifold learning at a local region.

Stabilisation

To perform stabilisation in the case of the paths obtained using manifold learning, we simply apply the inverse transformation of the motion estimate, applying the L1 optimal paths stabilisation would yield the same result. For the stabilisation of the original tracking motion estimate we apply the L1 optimal paths technique since the original tracking has errors.



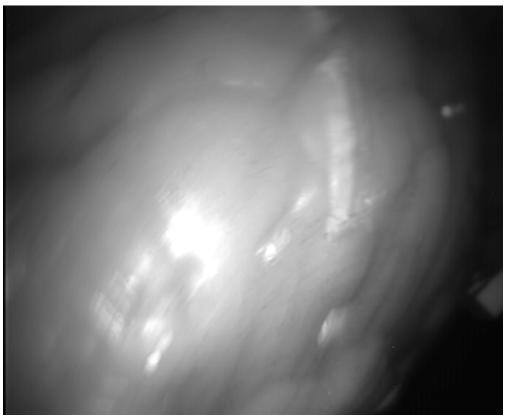
(a) Stabilised with manifold learning (Local)



(b) Stabilised with L1 optimal path using path estimate from tracking (Local)

(c) Stabilised with manifold learning (Global)

(d) Stabilised with L1 optimal path using path estimate from tracking (Global)



(e) Original

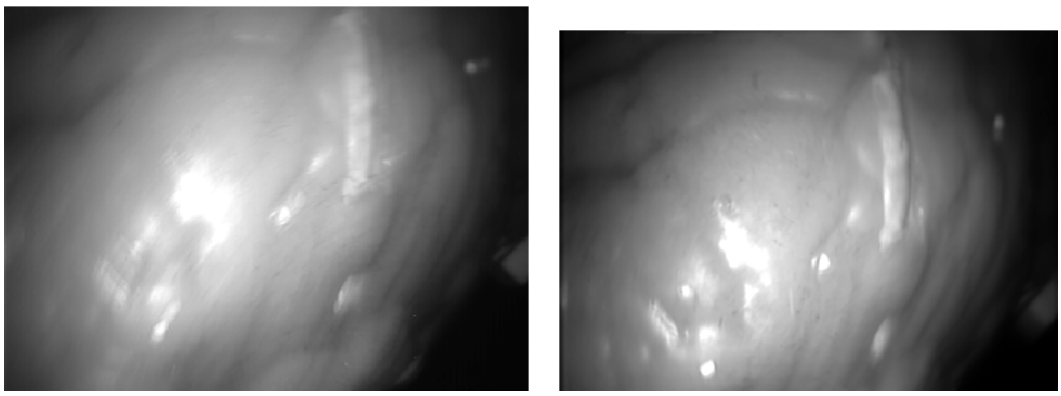
Figure 7.7: Mean images (with contrast adjustment) for 600 frames for the global and local case.

Tracking is performed on 600 frames. The mean images obtained for the stabilisation for both the local and global motion estimation mentioned previously are in figure 7.7. We see that the stabilisation obtained by using the path estimate from the manifold yields the best result both

in the global and local case. We see that in the case of (d) there is directional blurring in the y direction due to the drift mentioned in the previous chapter. The drift is less pronounced with the local motion estimation in (b) but nonetheless (a) is more sharp in the local tracked area indicating that the area remains perfectly stable.

Predicting stabilisation for new frames based on manifold

We use 300 frames as the training set for estimating a function of the x and y coordinates as a function of the embedded coordinates. We use $K = 3$ for the nearest neighbour. We then test the prediction on 600 new frames. We see that we obtain a good stabilisation result as attested by the mean images (figure 7.8), the inter-frame error (table 7.4) and the absolute error (table 7.5).



(a) Original (b) Obtained with predictive stabilisation

Figure 7.8: Mean images (with contrast adjustment) for 600 frames using predictive stabilisation based on manifold.

	600 frames	
	Original	Stabilised
Mean	24.4481	8.0656
Variance	163.9536	25.9227

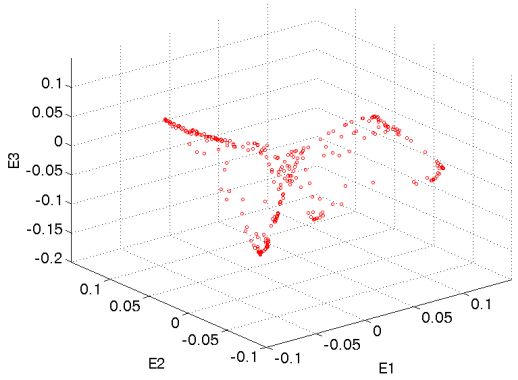
Table 7.4: Inter-frame error for 600 frames of the phantom heart using predictive stabilisation based on manifold learning.

	x direction	y direction
Mean	0.018	0.021
Variance	0.00029	0.00047

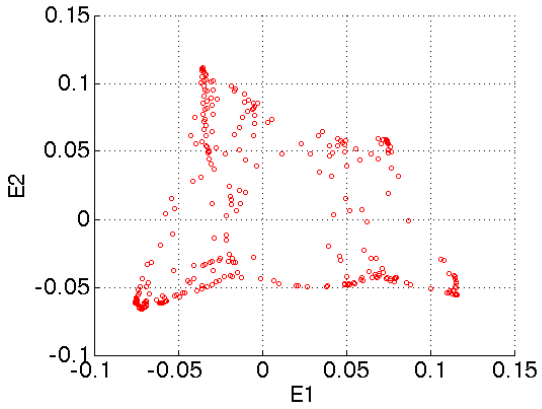
Table 7.5: Absolute error in position for the phantom heart video.

7.2.2 Real Heart

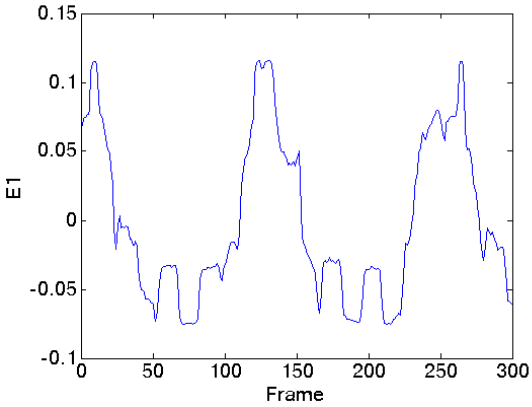
We perform dimensionality reduction for 300 frames using Laplacian Wigenmaps with 3 dimensions and 8 nearest neighbours. We use in-painting on the images so as to reduce the effect of the specular highlights. Figure 7.9 (a) shows the obtained manifold. (f) shows the evolution of the second component over number of frames without specular in-painting, we see when compared to (d) that specular highlights have a considerable impact.



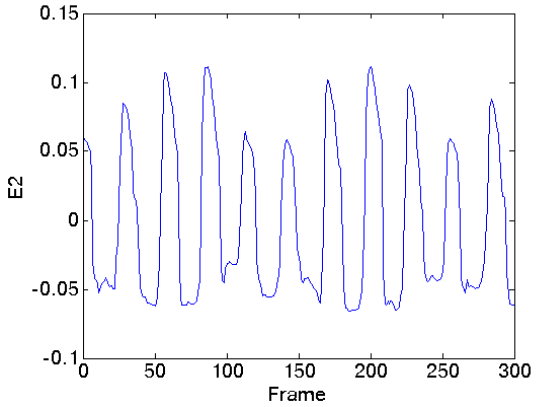
(a) 3D Manifold embedding



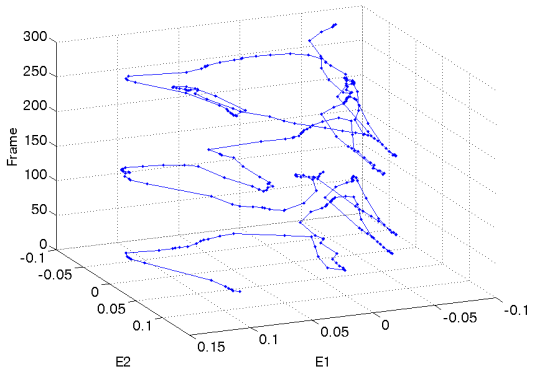
(b) 2D Manifold embedding



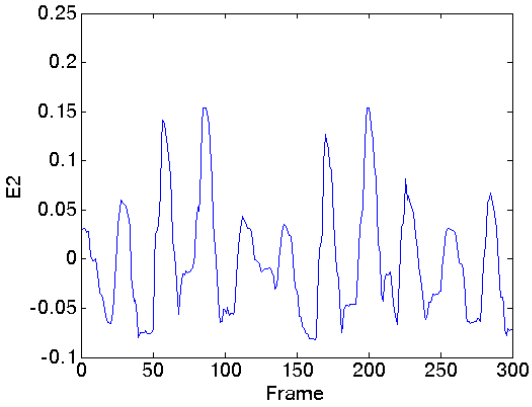
(c) Evolution over time for the first component $E1$



(d) Evolution over time for the second component $E2$



(e) Evolution over time

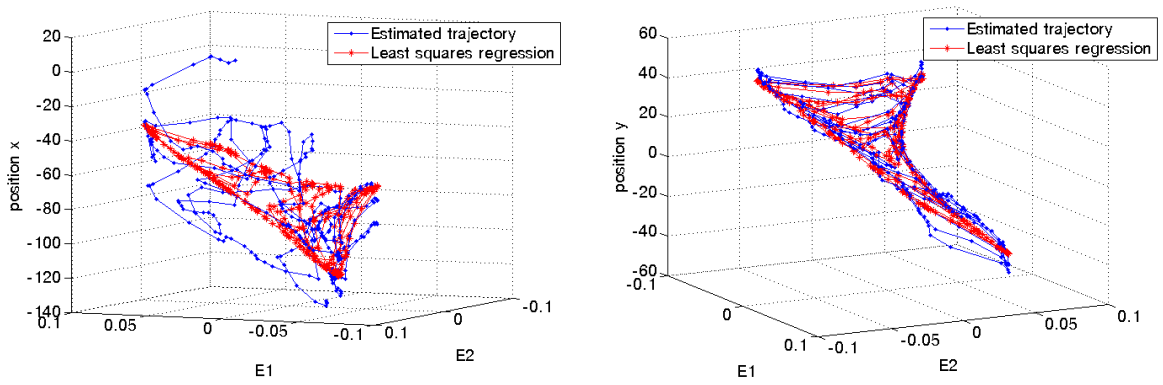


(f) Evolution over time of the second component E2 without specular in-painting for the images

Figure 7.9: Manifold for the real heart.

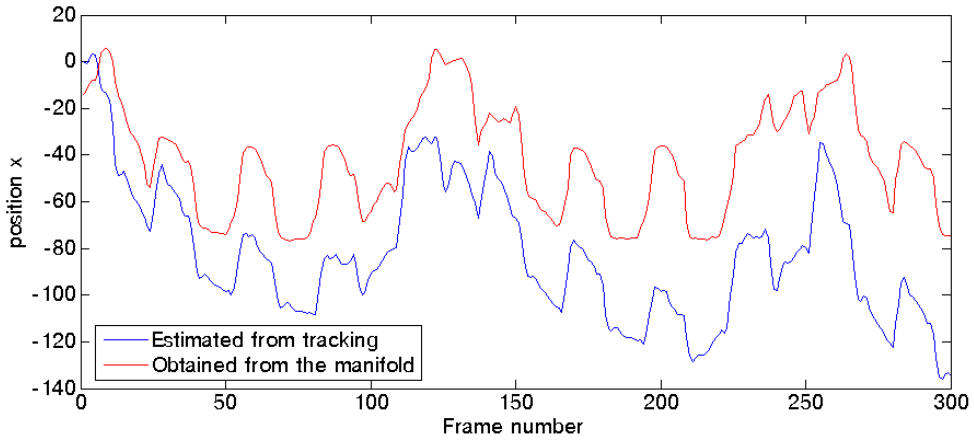
Trajectory Reconstruction

Global We estimate the translation motion globally in the original video by applying the Median Flow tracker on 300 frames. We then perform least squares regression to obtain an expression for the x and y coordinates as a linear function of the embedded coordinates $E1$, $E2$ and $E3$. We see the result of regression in red in figure 7.10 (a) and (b). Figure 7.11 (a) and (b) show the reconstructed paths in the x and y position obtained using the first 3 components. We see that overall the motion is similar apart from the drift in the motion estimate obtained from the tracking.

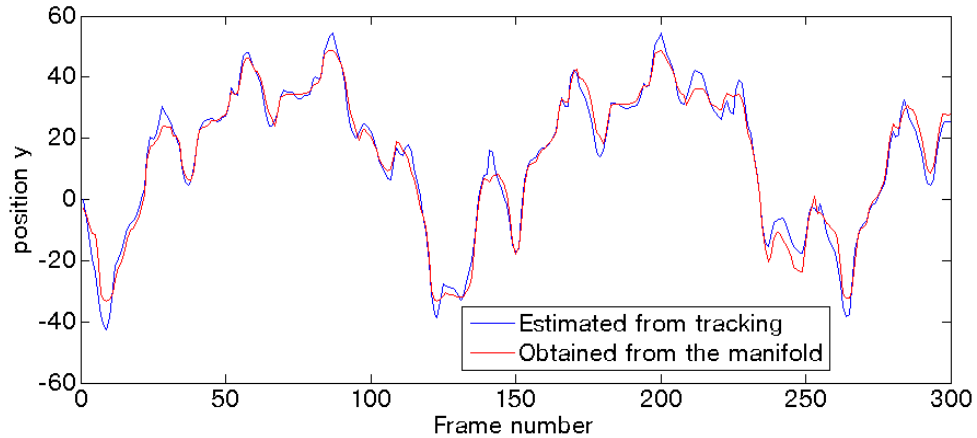


(a) (b)

Figure 7.10: Graphs showing the position as a function of the first two components E1 and E2 from the manifold. The estimated position from tracking of 300 frames is in blue. The red trajectory is the one obtained with regression.



(a) Estimated x position (blue) and the reconstructed x position (red) as a function of first 3 embedded coordinates.



(b) Estimated y position (blue) and the reconstructed y position (red) as a function of first 3 embedded coordinates.

Figure 7.11: Reconstructing x and y as functions of the embedded coordinates

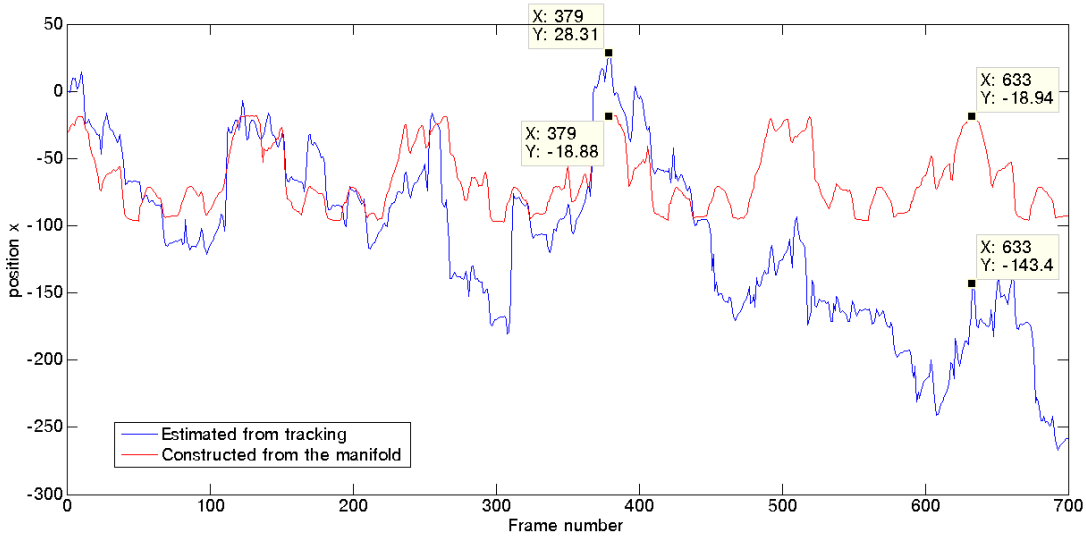
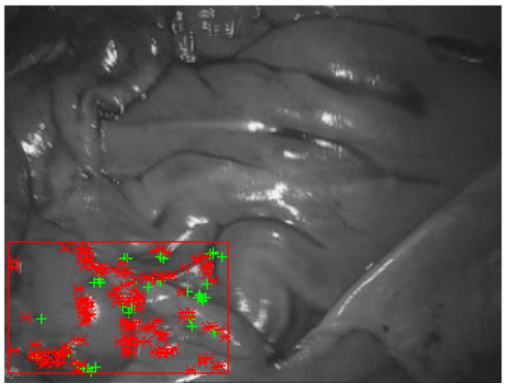
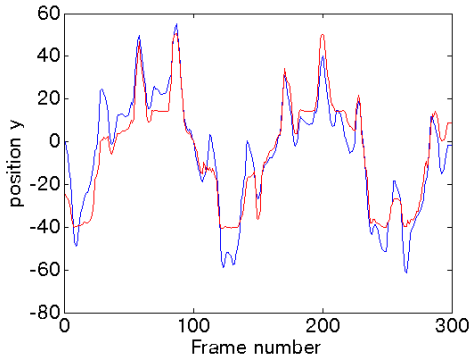


Figure 7.12: Graph showing the estimated path from tracking and the reconstructed path. Notice how there is no drift in the reconstructed path. For the same images at frame 379 and 633 shown in figure 6.27 they are in the same position according to the reconstructed path.

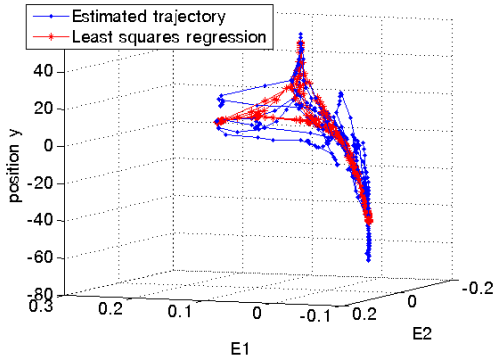
Local We perform local tracking on the bottom left hand side of the frame. The bounding box is fixed. We use the Median Flow tracker.



(a) Tracked area. The bounding box is fixed.



(b) Graph showing the position y over frames. Estimated from tracking (Blue). Reconstructed from the manifold (Red).



(c) Least squares regression for the y position.

Figure 7.13: Manifold learning for a local area.

Stabilisation

For the stabilisation, to see the effect over longer periods, we carry out the tracking on 700 frames in the global case.

Looking at the mean images obtained for the global stabilisation in figure 7.14, we that the one obtained using manifold learning has the most distinct features. However when we look at the inter-frame error in table 7.6, we see that it is slightly lower in the one obtained using the L1 optimal paths stabilisation. This is most likely due to the fact that by using the original tracked estimate there is a bit more detail about small motion which is compensated for, something that is not present in the reconstructed path using the manifold.

	700 frames		
	Original	Stabilised (L1 optimal paths)	Stabilised (manifold)
Mean	72.036310	43.224052	48.832651
Variance	896.570364	565.865825	595.015015

Table 7.6: Inter-frame error for the whole frame (Global).

(a) Stabilised with manifold learning

(b) Stabilised with L1 optimal camera paths using the estimated tracked path.

(c) Original

Figure 7.14: Mean images (with contrast adjustment) for 700 (Global)

For the local stabilisation we see that in the bottom left corner of the mean images in figure 7.15, the one obtained with manifold learning yields the best result. However in the inter-frame error (table 7.7), the one obtained using the manifold is higher, which could be explained by the fact it was calculated on the whole image and not in the local area on which we carried out the stabilisation. This also shows that the inter-frame measure is merely indicative and does not fully represent the perceived stabilisation.

	300 frames		
	Original	Stabilised (L1 optimal paths)	Stabilised (manifold)
Mean	71.9481	49.7140	59.7316
Variance	851.2438	790.5801	824.4883

Table 7.7: Inter-frame error for the whole frame (Local).

(a) Stabilised with manifold learning

(b) Stabilised with L1 optimal paths using original tracked estimate

(c) Original

Figure 7.15: Mean images (with contrast adjustment) for 300 (Local).

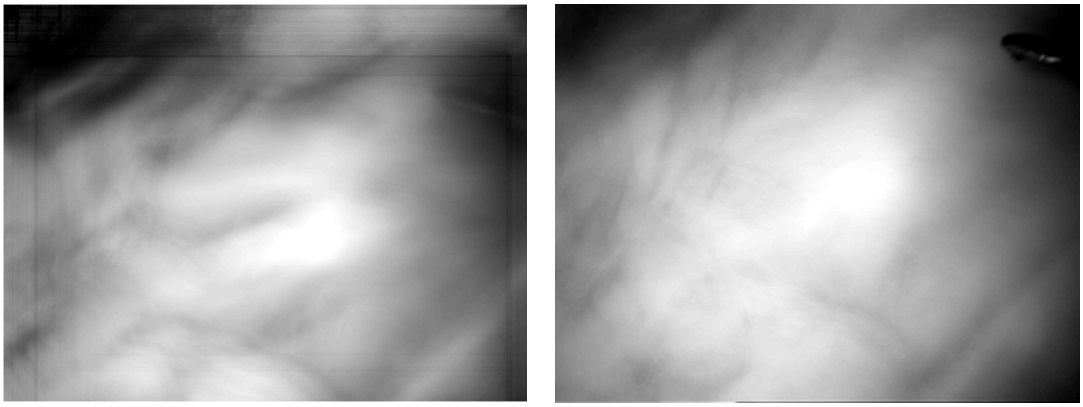
Predicting Stabilisation

We perform stabilisation using manifold learning on 700 frames. We use the first 500 frames of the video as the training set i.e. the first 500 frames are used for the regression. We then perform predictive stabilisation for last 200 frames. To test the values obtained from the prediction, we also perform stabilisation using all the 700 frames to perform the regression in order to obtain the absolute error. We use $K = 7$ for the nearest neighbour.

In figure 7.16 we see that there is a reduction in motion whereby it is possible to see some features in the image. Table 7.8 shows the mean and variance of the absolute error of the x and y direction. Figure 7.17 shows the absolute error in the y direction over frame. We see that this time we have a considerably less amount of error in the absolute error.

	x direction	y direction
Mean	0.4612	0.4225
Variance	0.3347	0.3906

Table 7.8: Absolute error in position for the real heart video.



(a) Stabilised with manifold learning (b) Original

Figure 7.16: Mean images (with contrast adjustment) for 300 using predictive stabilisation (Global).

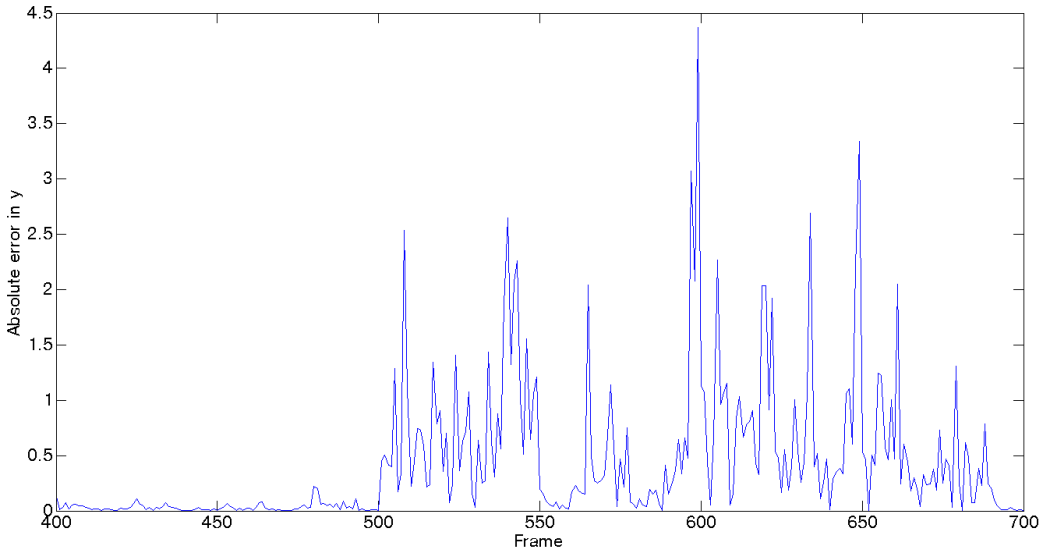


Figure 7.17: Absolute error in the y direction for the real heart video using predictive stabilisation based on nearest embedded coordinates.

7.2.3 Evaluation

We have seen that it is possible to obtain a good overall motion estimation using manifold learning. We have performed motion estimation by tracking on an initial set of frames and computed the corresponding manifold. We have found that we can express the x y coordinates as a linear function of the embedded coordinates. The motion estimate obtained this way can be used to perform stabilisation by simply applying the inverse of the motion estimate.

In the case of the phantom heart we have seen that we can obtain good global and local stabilisation using the manifold learning technique. Although we can also obtain good stabilisation using the L1 optimal paths, however it suffers from the problem of the accumulation of errors in the motion estimate.

In the case of the real heart we see that using manifold learning for stabilisation yields a better outcome whereby there is roughly no deviation over time due to an improved motion estimate. Using the original motion estimate with the L1 paths algorithm, which seems to yield a lesser inter-frame error than the case of manifold learning, is also capable of stabilising the video, however there is a residual amount of motion which is seen in the mean image when attempting to track over longer periods of time.

The manifold learning technique seems to offer a better solution when used with a translation

model whereby it uses directly the motion estimate based on the manifold to perform the stabilisation by simply using the inverse to counteract the motion. Using the inverse is not possible with the motion estimate obtained from tracking, as due to the errors in the motion estimate, the image would eventually end up going out of frame.

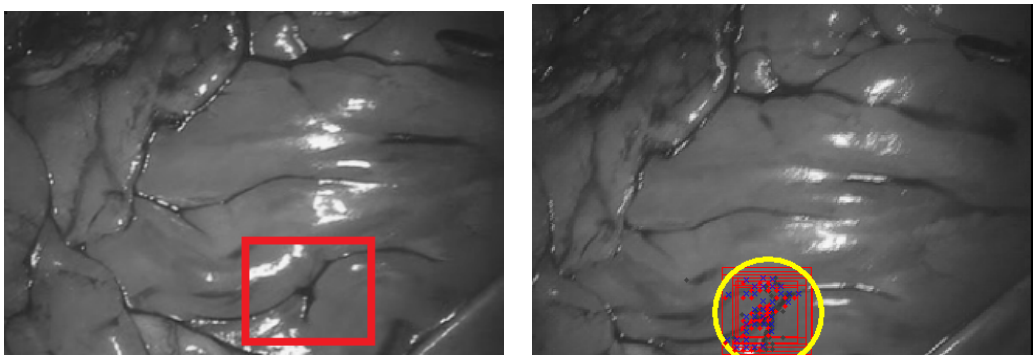
We also saw that in the case of manifold learning, the predictive stabilisation yields much less error than using directly the the update transforms as in experiment 4. However we should note that we used an affine model in experiment 4 and not a translation model which could have yielded a lower error.

A limitation for the predictive stabilisation based on the nearest neighbour approach is that it is computationally expensive when used globally as we are comparing entire images. Using it on a smaller sized image or locally can be more efficient. Also we should note that the specular highlights in-painting technique can also be computationally expensive depending on the size of the image.

7.3 Experiment 6: Tracking and Stabilising with TLD

For this experiment we briefly look the possibility to use the TLD tracker to perform real-time tracking and stabilisation for medical image sequences. We select an area for tracking which can be seen in figure 7.18. Tracking is performed twice on an initial set of 300 frames so as to increase the chances of the detector learning and then tracking is carried on the entire video of 1000 frames. Figure 7.19 shows the x and y coordinates of the centre of the bounding box, we see that the tracker fails at some frames (indicated by the lack of connecting lines), this is due to the change incurred to the area due to the contractions.

The result of simply cropping the bounding box at each frame to perform stabilisation is show as a mean image in figure 7.20. We see that the result obtained suggests a good stabilisation result whereby even in the mean image obtained with 1000 frames we can still distinguish the tracked area, blurring is inevitable due to the surface deformations.



(a) The area used for tracking.

(b) The Red bounding boxes indicate possible matches according to the detector. The yellow circle indicates the position.

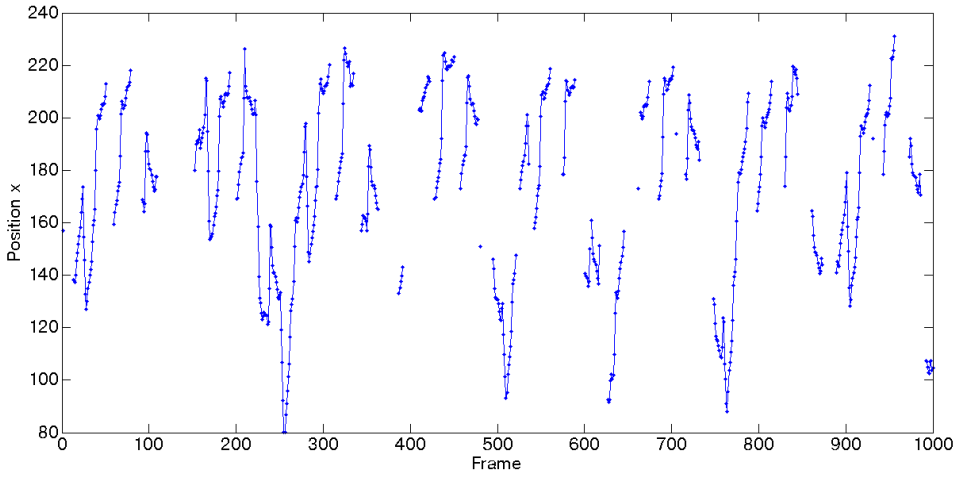
Figure 7.18: Tracking using TLD

The reason why the TLD tracker is capable to perform well is that it is able to adjust the position of the tracker by the addition of the adaptive detector.

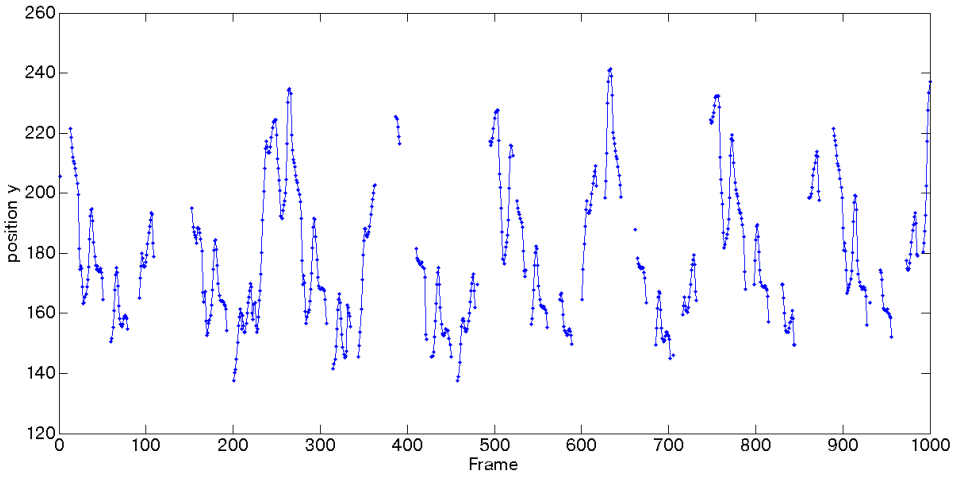
Improving Performance

We could also note that, given that in the scene the motion is repetitive then it is possible to increase the performance of the tracker by limiting the search space of the detector since it works by scanning the whole image with a sliding window.

To reduce the search space we could perform the tracking on a smaller search area of the frame after an initial tracking phase to determine the maximum distances covered in the x and y

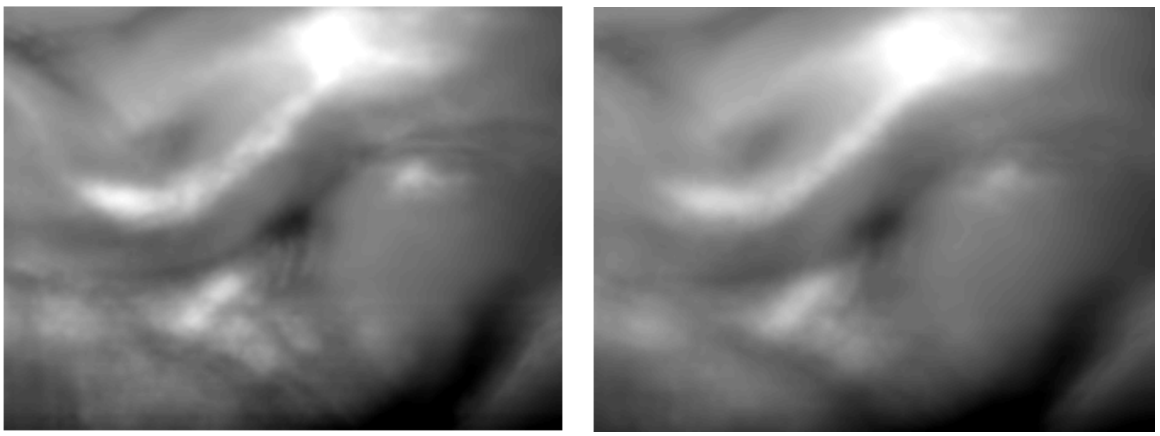


(a) The x position of the centre of the bounding box. The locations where there are no connecting lines is where the tracker and detector failed.



(b) The y position of the centre of the bounding box. The locations where there are no connecting lines is where the tracker and detector failed.

Figure 7.19: The x and y coordinates of the detected object over time.



(a) 300 frames.

(b) 1000 frames

Figure 7.20: Mean images (with contrast adjustment, without specular in-painting).

directions.

An alternative method would be to implement a prediction scheme to determine the area to

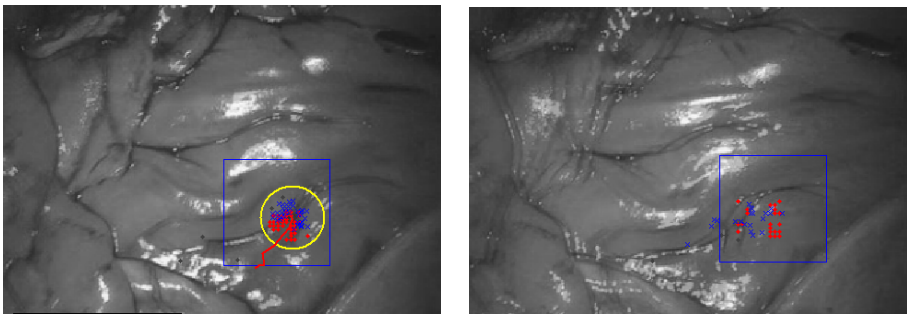
scan. We tested to see if it would work:

We repeat the tracking on an initial set of frames to get a better detection performance until there was relatively a small amount of tracking failures, the missing locations are interpolated. After that we determine the maximal amount of displacement between two consecutive frames in order to determine a size for the search area box (we call this training phase). For the rest of the video, the search box would move according to the position of the location of the object, determining the search area for the next frame.

In the eventual case when the tracking fails, then the search box would still continue moving according to the closest obtained path during the training phase. We determine this simply by finding the closest segment match, in the paths obtained during training, of the segment just before the tracking failed. The position of the search box would then move according to next locations starting from the closest match found.

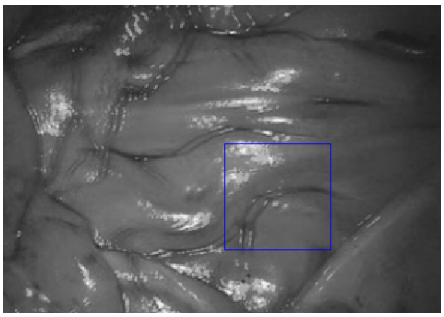
We have implemented this as a test and found that it does lead to an considerable improvement in performance. Figure 7.21 shows frames from the tracking, the blue box is the reduced search area. The tracking failed at frame 789 (b) the blue box continues to follow according to the previous closest encountered trajectory from the training phase. The box deviates a bit in frame 799 (c). (e) shows the re-detection. We have used here a small search box, we could consider using a slightly larger search box to deal with situations like in (d).

Other methods might be more adapted for reducing the search area, but we have found that it is possible to increase performance but adopting such prediction schemes.

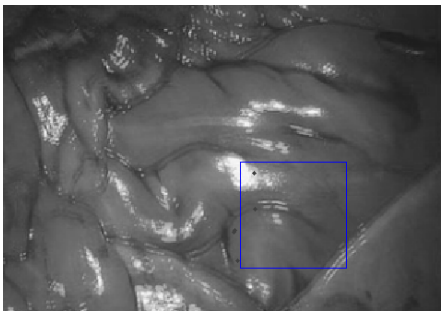


(a) Frame 788

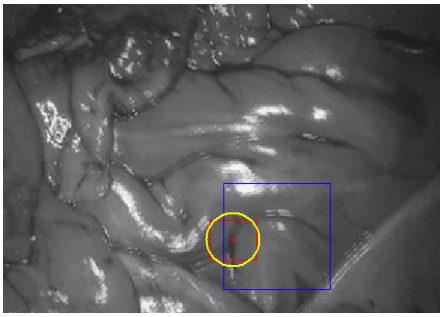
(b) Frame 789



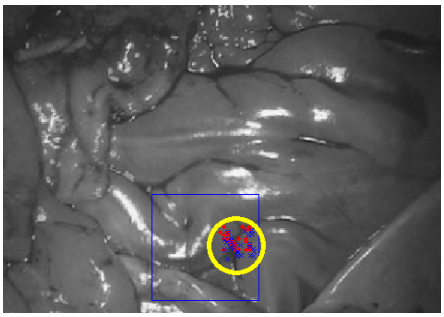
(c) Frame 790



(d) Frame 799



(e) Frame 800



(f) Frame 801

Figure 7.21: Search area prediction scheme for TLD tracker. Blue box indicates search area.

7.4 Conclusion

In experiment 4 we have seen that using the nearest neighbour approach to perform predictive stabilisation using precomputed update transforms did not yield a satisfactory result. It would give a good result in the case of the phantom heart if there were no error in the motion estimate. In the case of the real heart however, the problem is still further exacerbated due to the type of motion of the cardiac surface. Alternative means for extending the L1 optimal paths stabilisation algorithm need to be found.

In experiment 5 we have seen that performing motion estimation and stabilisation using manifold learning seems to give good results somewhat comparable to those obtained using the L1 optimal paths algorithm. The advantage in the manifold learning case is that it is possible to obtain an estimate for the overall motion with there being no drift / error in the motion estimate. The L1 optimal paths technique would yield better results if the motion estimate were more accurate.

We have only used three embedded coordinates from the manifold, we have tried using more but it did not lead to an improvement and in some cases the results were worse.

The technique using manifold learning used in a post-processing setting gives good results, however for predictive stabilisation, even though we have obtained satisfactory results, the approach is limited in the sense the scene has to be predicable. Using it in the real setting were different and sometimes unpredictable situations would cause the method to fail, a simple case would be if the camera were to move then it would be impossible to calculate the nearest neighbour accurately.

Stabilisation using adaptive trackers such as TLD, as briefly seen in the last experiment, seems to offer a possible alternative solution for performing stabilisation and should require further investigation.

Chapter 8

Conclusions and Future Work

8.1 Conclusions

The motivation behind this project is to investigate 2D virtual motion stabilisation techniques for their effectiveness at compensating cardiac and respiratory motion in medical image sequences acquired during MIS interventions as it would be greatly of aid to the surgeon.

The aim of this project was to test the effectiveness of the current state of the art 2D stabilisation technique proposed by Grundmann et al. [1] when applied to the medical image sequences containing cardiac and respiratory motion. We also aimed at exploring alternative methods by which we could perform stabilisation. We have looked at using Laplacian Eigenmaps to learn the manifold from the video and find a linear mapping from the embedded coordinates to the camera path coordinates. The mapping obtained can then be used to perform stabilisation on the rest of the video.

We have successfully implemented both stabilisation techniques in MATLAB as well as a modified specular highlight detection and in-painting method.

In a first set of experiments we have tested the performance of the stabilisation algorithm on three videos, and have seen that there is reduction in motion.

The first video was a hand-held video with considerable shake containing a rigid background, the type of video the stabilisation algorithm was designed for. The end result of the stabilisation is a video with a jitter free smooth camera path.

The second video was that of a phantom heart with a static camera, applying the stabilisation algorithm with an affine model produces a video which eliminates the cardiac motion. However due to an accumulation of errors in the motion estimation, the end result has a slow directional drift which can be corrected.

For the third experiment we applied the algorithm to a video of the real heart, we have noticed that there is a noticeable reduction in the motion from the cardiac and respiratory cycle, however due to the complicated nature of the surface movement, the obtained global motion estimate is not very accurate leading to the persistence of some residual motion. Nonetheless, the result is satisfactory.

In the second set of experiments we looked at performing predictive stabilisation based on the nearest neighbour approach. We also looked at alternative possible means for stabilisation.

In the forth experiment we have found that extending the L1 optimal paths algorithm into a simple predictive stabilisation scheme based on K nearest neighbours does not lead to a satisfactory stabilisation result. Alternative means to extend the algorithm should be investigated.

In the fifth experiment we have investigated whether it is possible to perform post and predictive stabilisation based on manifold learning. We have found that a stabilisation based on a translation model is achievable and yields good overall stabilisation results. Using just three embedded coordinates does not compensate for fine motion detail however. The predictive stabilisation scheme based on K nearest neighbour also leads to good results overall, but the technique

is computationally expensive if applied globally.

In the last experiment we have briefly looked at the performance of an adaptive tracker called TLD, we have seen that it is possible to achieve real-time stabilisation by tracking a specific area. It is also possible to increase the real-time performance by adopting a prediction scheme. However the tracker fails occasionally to detect the object even though it is still present in the scene due to the rotation induced by the contractions. The limitation is identified by the authors [31]. Nonetheless the tracker shows potential and should be further investigated.

8.2 Future Work

For future work we see two possible directions. One based on using manifold learning as a means to perform stabilisation and the other is using local motion estimation by tracking landmarks using adaptive trackers such as TLD.

With the manifold learning technique, we have seen that it is possible to achieve local stabilisation on different areas in the frame by computing the local manifold of that area. Exploring the possibility of using hierarchical manifold learning [29] to obtain motion estimates in a hierarchical manner seems like a good possibility. Having motion estimates for each of the local areas could make it possible to actively change the stabilised area or to improve the global stabilisation. We have only applied the technique to a translation model, one could also explore if it is possible to obtain a motion model with higher DOF by using the motion estimate of all of the tracked areas.

We could also consider trying different manifold learning techniques and different distance measures. Also we have modelled the relation between the x,y coordinates and the embedded coordinates as linear, which turns out to yield good results, nonetheless we could still possibly explore using a dynamical motion model instead as in [26].

With the TLD tracker, a possible direction would be looking for means to make it more suited at detecting landmarks on deformable tissue and perhaps implementing multiple object trackers to track different landmarks in the scene such that if tracking on one of landmarks fails then it is possible to continue tracking from the other landmarks.

Lastly, Another thing one could possibly explore would be the challenging task of finding the means of extending the L1 optimal paths technique into real-time.

Bibliography

[1] Matthias Grundmann, Vivek Kwatra, and Irfan A. Essa. Auto-directed video stabilization with robust L1 optimal camera paths. In *CVPR*, pages 225–232. IEEE, 2011.

[2] Mirko Arnold, Anarta Ghosh, Stefan Ameling, and Gerard Lacey. Automatic segmentation and inpainting of specular highlights for endoscopic imaging. *EURASIP Journal on Image and Video Processing*, 2010(1):814319, 2010.

[3] Y. Matsushita, E. Ofek, Weina Ge, Xiaoou Tang, and Heung-Yeung Shum. Full-frame video stabilization with motion inpainting. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(7):1150–1163, 2006.

[4] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

[5] Feng Liu, Michael Gleicher, Hailin Jin, and Aseem Agarwala. Content-preserving warps for 3d video stabilization. *ACM Trans. Graph.*, 28(3):44:1–44:9, July 2009.

[6] P. Mountney, D. Stoyanov, and Guang-Zhong Yang. Three-dimensional tissue deformation recovery and tracking. *Signal Processing Magazine, IEEE*, 27(4):14–24, 2010.

[7] Peter Mountney, Benny Lo, Surapa Thiemjarus, Danail Stoyanov, and Guang Zhong-Yang. A probabilistic framework for tracking deformable soft tissue in minimally invasive surgery. In *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention*, MICCAI'07, pages 34–41, Berlin, Heidelberg, 2007. Springer-Verlag.

[8] Stamatia Giannarou, Marco Visentini Scarzanella, and Guang-Zhong Yang. Probabilistic Tracking of Affine-Invariant Anisotropic Regions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(1):130–143, 2013.

[9] Danail Stoyanov, George P. Mylonas, Fani Deligianni, Ara Darzi, and Guang-Zhong Yang. Soft-Tissue Motion Tracking and Structure Estimation for Robotic Assisted MIS Procedures. In James S. Duncan and Guido Gerig, editors, *MICCAI (2)*, volume 3750 of *Lecture Notes in Computer Science*, pages 139–146. Springer, 2005.

[10] Danail Stoyanov and Guang-Zhong Yang. Stabilization of image motion for robotic assisted beating heart surgery. In *Proceedings of the 10th international conference on Medical image computing and computer-assisted intervention - Volume Part I*, MICCAI'07, pages 417–424, Berlin, Heidelberg, 2007. Springer-Verlag.

[11] Rogério Richa, Philippe Poignet, and Chao Liu. Efficient 3d tracking for motion compensation in beating heart surgery. In *Proceedings of the 11th International Conference on Medical Image Computing and Computer-Assisted Intervention, Part II*, MICCAI '08, pages 684–691, Berlin, Heidelberg, 2008. Springer-Verlag.

[12] Min Tang, Yuan-Quan Wang, PhengAnn Heng, and De-Shen Xia. Tag stripes tracking from cardiac mri by bayesian theory. In Guang-Zhong Yang and Tian-Zi Jiang, editors, *Medical Imaging and Augmented Reality*, volume 3150 of *Lecture Notes in Computer Science*, pages 245–252. Springer Berlin Heidelberg, 2004.

[13] T. Ortmaier, Martin Groger, D.H. Boehm, V. Falk, and G. Hirzinger. Motion estimation in beating heart surgery. *Biomedical Engineering, IEEE Transactions on*, 52(10):1729–1740, 2005.

[14] Haytham Elhawary and Aleksandra Popovic. Robust feature tracking on the beating heart for a robotic-guided endoscope. *The International Journal of Medical Robotics and Computer Assisted Surgery*, 7(4):459–468, 2011.

[15] J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pages 593–600, 1994.

[16] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[17] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *In European Conference on Computer Vision*, pages 430–443, 2006.

[18] D.G. Lowe. Object recognition from local scale-invariant features. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1150–1157 vol.2, 1999.

[19] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Comput. Vis. Image Underst.*, 110(3):346–359, June 2008.

[20] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.

[21] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. pages 674–679, 1981.

[22] Bernd Jahne, Peter Geissler, and Horst Haussecker, editors. *Handbook of Computer Vision and Applications with Cdrom*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1st edition, 1999.

[23] Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A Global Geometric Framework for Nonlinear Dimensionality Reduction. *Science*, 290(5500):2319–2323, December 2000.

[24] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems 14*, pages 585–591. MIT Press, 2001.

[25] Robert Pless and Richard Souvenir. A Survey of Manifold Learning for Images. *IPSJ Transactions on Computer Vision and Applications*, 1:83–94, 2009.

[26] Ali Rahimi and et al. Learning appearance manifolds from video. In *IN COMPUTER VISION AND PATTERN RECOGNITION (CVPR*, pages 868–875. CVPR, 2005.

[27] Christian Wachinger, Mehmet Yigitsoy, Erik-Jan Rijkhorst, and Nassir Navab. Manifold learning for image-based breathing gating in ultrasound and {MRI}. *Medical Image Analysis*, 16(4):806 – 818, 2012.

[28] Qilong Zhang, Richard Souvenir, and Robert Pless. Segmentation informed by manifold learning. In Anand Rangarajan, Baba Vemuri, and AlanL. Yuille, editors, *Energy Minimization Methods in Computer Vision and Pattern Recognition*, volume 3757 of *Lecture Notes in Computer Science*, pages 398–413. Springer Berlin Heidelberg, 2005.

[29] Kanwal K. Bhatia, Anil Rao, Anthony N. Price, Robin Wolz, Jo Hajnal, and Daniel Rueckert. Hierarchical manifold learning. volume 7510 of *Lecture Notes in Computer Science*, pages 512–519. Springer Berlin Heidelberg, 2012.

[30] Z. Kalal, K. Mikolajczyk, and J. Matas. Forward-backward error: Automatic detection of tracking failures. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 2756–2759, 2010.

[31] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[32] JungHwan Oh, Sae Hwang, JeongKyu Lee, Wallapak Tavanapong, Johnny Wong, and Piet C. de Groen. Informative frame classification for endoscopy video. *Medical Image Analysis*, 11(2):110 – 127, 2007.

[33] M. Niskanen, O. Silven, and M. Tico. Video stabilization performance assessment. In *Multimedia and Expo, 2006 IEEE International Conference on*, pages 405–408, 2006.

[34] S. Chikkerur, V. Sundaram, M. Reisslein, and L.J. Karam. Objective video quality assessment methods: A classification, review, and performance comparison. *Broadcasting, IEEE Transactions on*, 57(2):165–182, 2011.

Appendix

Running the Code

In order for the full implementation to run, the following are required:

- MATLAB R2013a+
- OpenCV 2.4+

Running the main GUI in MATLAB can be done by executing the file *stabilization_gui.m*.

Compiling the C++ code can be done through MATLAB by running the *compile.m* file, however a compiler has to be linked with MATLAB. Instructions on how to do that in MATLAB can be obtained by typing the following *mex - help*. Typing *mex -setup* allows the user to select or change the compiler configurations.

A readme file is provided with the code to give additional instructions.