January 2010

# On Motion Parameterizations in Image Sequences from Fixed Viewpoints

Manfred Georg
*Washington University in St. Louis*

WASHINGTON UNIVERSITY IN ST. LOUIS

School of Engineering and Applied Science

Department of Computer Science and Engineering

Dissertation Examination Committee:
Robert Pless, Chair
Guy Genin
Cindy Grimm
Tao Ju
William Richard
John Shareshian

# ON MOTION PARAMETERIZATIONS IN IMAGE SEQUENCES FROM FIXED VIEWPOINTS

by

Manfred Georg

A dissertation presented to the
Graduate School of Arts and Sciences
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

August 2010

Saint Louis, Missouri

ABSTRACT OF THE THESIS

On Motion Parameterizations in Image Sequences from Fixed Viewpoints

by

Manfred Georg

Doctor of Philosophy in Computer Science

Washington University in St. Louis, 2010

Research Advisor: Professor Robert Pless

This dissertation addresses the problem of parameterizing object motion within a set of images taken with a stationary camera. We develop data-driven methods across all image scales: characterizing motion observed at the scale of individual pixels, along extended structures such as roads, and whole image deformations such as lungs deforming over time. The primary contributions include (a) fundamental studies of the relationship between spatio-temporal image derivatives accumulated at a pixel, and the object motions at that pixel, (b) data driven approaches to parameterize breath motion and reconstruct lung CT data volumes, and (c) defining and offering initial results for a new class of Partially Unsupervised Manifold Learning (PUML) problems, which often arise in medical imagery.

Specifically, we create energy functions for measuring how consistent a given velocity vector is with observed spatio-temporal image derivatives. These energy functions are used to fit parametric snake models to roads using velocity constraints. We create an automatic data-driven technique for finding the breath phase of lung CT scans

which is able to replace external belt measurements currently in use clinically. This approach is extended to automatically create a full deformation model of a CT lung volume during breathing or heart MRI during breathing and heartbeat. Additionally, motivated by real use cases, we address a scenario in which a dataset is collected along with meta-data which describes some, but not all, aspects of the dataset. We create an embedding which displays the remaining variability in a dataset after accounting for variability related to the meta-data.

# Acknowledgments

I would like to thank everyone who taught and helped me during my studies. Together you have shaped me into the person I am today, and I am eternally grateful. In particular, I thank my advisor, Robert Pless, who has always been insightful, more helpful than I deserve, and a deep inspiration. I thank Richard Souvenir, Andrew Hope, and Nathan Jacobs for their many discussions and collaborations. I thank Jonathan Cannon for first implementing some of the 4D CT lung code with which I worked. Finally, I thank my committee members for their direction and support.

On a personal note, I thank my family and the many friends who have enormously enriched my graduate school experience. Charles Comstock in particular has been the greatest friend I can imagine: I will fondly remember our late night conversations. Sarah Boyle deserves special thanks for supporting and putting up with me during my studies.

This dissertation is dedicated to my mother, who prodded and supported me in every way possible, and to my father, whose presence I inevitably feel whenever I write an equation.

Manfred Georg

*Washington University in Saint Louis*
*August 2010*

# Contents

# List of Figures

# Chapter 1

# Introduction

The intelligent analysis of image and video sequences has become an important part of an increasing number of applications. For example, portable cameras have integrated face detectors, photo archiving tools make use of object recognition and image search, and medical image analysis requires segmentation and registration methods.

With increasing processing power, many methods which have been used previously only on single images can now be applied to entire video sequences. Large amounts of video data are collected for the analysis of motion patterns. Surveillance, anomaly detection, target tracking, and action recognition are all examples of applications which use video data. In medical imaging, methods for determining tissue motion are both becoming integrated into patient diagnosis and treatment systems, and used for research purposes.

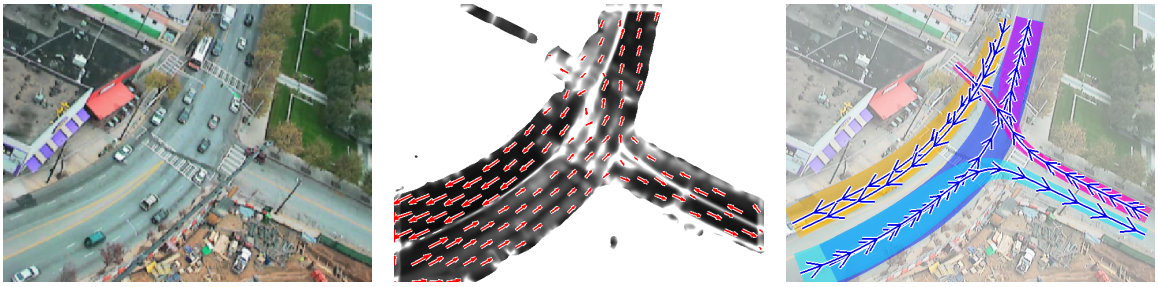


Figure 1.1: Left: One frame from a video of traffic moving over roads. Middle: A vector field extracted from the motion patterns observed at each pixel over the length of the video. Right: A parameterization of the motion patterns in the scene using snakes.

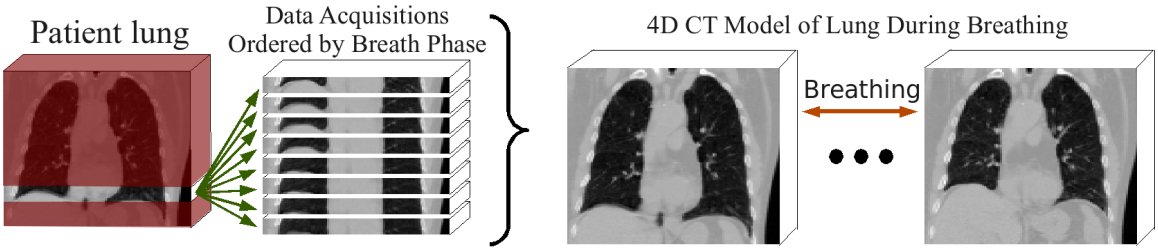



Figure 1.2: Left: Multiple images of a part of the lung are acquired and breath phase is assigned to each acquisition through analysis of the data volume. Right: A full 4D CT lung model is created which captures both the spatial appearance of the lung and the dynamics of how tissue moves during breathing.

A large number of applications make use of a stationary camera which observes the same scene over a long period of time. This simplifies the problem of scene understanding, since, in this setting, algorithms do not need to compensate for the movement of the camera. This allows for more powerful and detailed analysis than in the general case of a moving camera. Furthermore, some questions can only be answered by observing a scene for a long period of time. This dissertation will address several different scenarios in which stationary cameras observe a scene with moving objects of interest. In each problem domain we will seek to understand the motion within the scene.

We have organized each chapter as a self contained unit with its own related work, technical contributions, and evaluation sections. Chapter 2 studies video sequences from stationary cameras looking at consistent patterns of motion, such as traffic over roads. We investigate different ways of extracting a motion vector field from summary statistics of motion at each pixel. We extend the use of these summary statistics to parameterize the motion within the scene using snake models. Figure 1.1 shows an example frame from a video sequence, an extracted motion vector field, and the final parameterization of motion within the scene.

Chapter 3 studies anatomical tissue motion within medical images, particularly lung tissue motion in 4D CT lung datasets. Our datasets are essentially image sequences from a stationary camera looking at a periodic action, either heartbeat or breathing. To create a coherent model of the tissue volume, the breath or heartbeat phase must be determined for each image acquisition. Once the phase of each acquisition is found, a full deformation model can be constructed to model the entire tissue volume as it


Figure 1.3: Top: Images of a kestrel wing flap cycle, this is a short term variation which we are able to parameterize using PUML methods. Bottom: Images showing the long term variations in rest wing position, tail variation, and general body pose which we are able to automatically ignore, given the timestamps of frames in the video sequence.

changes during physiological activity. Figure 1.2 shows CT image acquisitions from a lung patient, which are ordered by breath phase and then synthesized into a complete motion model of the lung.

Motivated by our work in medical imaging, where extra meta-data such as breath phase is frequently available from an external data source, in chapter 4, we investigate the more theoretical question of how to create improved embeddings of datasets when information partially explaining the variation in the dataset is provided. We formulate the problem of Partially Unsupervised Manifold Learning (PUML) in which we seek to embed a high dimensional dataset into a low dimensional space, preserving as much structure as possible and simultaneously ensuring that the output embedding is statistically independent of a given known parameter. Ensuring independence of the output from the known parameter means that each new parameter provides additional information about the causes of variation in the dataset. Our methods will allow us to use the timestamps of frames in a video to ignore long term variations in a dataset and more effectively parameterize short term variations. For example, in figure 1.3 we will automatically parameterize the short term variation of wing flaps and ignore the long term variations in rest body position, tail position, and wing positions.

**Contributions:** We list the 5 main contributions within this dissertation and give a brief summary of each.

1. **A general study of the properties of spatio-temporal derivatives accumulated over time at a single pixel in a video.** We develop and evaluate 3 different energy functions for determining the consistency of a velocity vector with a spatio-temporal structure tensor. We also analyze the temporal equivalent to the aperture problem, and find the standard use of total least squares is less appropriate for estimating motion in this setting. This contribution is presented in chapter 2.

2. **An explicit derivation of energy functions for fitting snake models by velocity to spatio-temporal derivatives.** This allows for stable open-ended snakes, a configuration which is not possible using traditional snake models. We derive analytical derivatives of the snake energy functions, enabling efficient optimization algorithms. This contribution is presented in chapter 2.

3. **A novel, data driven method of computing the breath phase of data acquisitions in a 4D CT lung dataset.** We demonstrate the ability to produce smoother reconstructions than the industry standard method which additionally requires measurements from an external belt. This contribution is presented in chapter 3.

4. **A novel algorithm for modeling deformable tissue motion.** We model tissue using a reference volume and deformation map and are able to simultaneously solve for the object state, deformation map, and reference volume. We are able to improve the smoothness of 4D CT lung reconstructions while also providing a motion model of tissue within the lung during breathing. This contribution is presented in chapter 3.

5. **A formulation and several solution approaches to a new problem, Partially Unsupervised Manifold Learning (PUML).** We give automated algorithms to ignore long term trends in datasets and improve the parameterization of short term trends. We are able to embed manifolds while ignoring variation in the dataset due to cyclic known parameters. This contribution is presented in chapter 4.

# Chapter 2

# Flow Constraints and Fitting Snakes by Velocity


## Abstract

*We consider the problem of estimating consistent motion patterns from video data using spatio-temporal image intensity derivatives. We explore the failure modes unique to motion estimation using spatio-temporal derivatives through time. We illustrate different ways to use the spatio-temporal structure tensor to define energy functions over flow vectors which express how consistent those vectors are with observed image derivatives. We show that unlike in the standard optic flow problem, it is a poor choice to use total least squares to solve for motion in a time series. We define novel energy functions to fit B-Spline snakes to roads by aligning their direction and speed to be maximally consistent with the spatio-temporal derivatives. Due to the constraint on the derivative of the snake, this produces stable, open-ended snakes, without specifying the location of endpoints. This completes a system to make a snake based parameterization of the position, direction, speed, and width of roads.*


## 2.1 Introduction

For many video applications, data is acquired in order to understand the motion patterns within the scene. In video surveillance and video-microscopy, video is frequently acquired from a stationary or georegistered camera, so motion patterns continue to

affect the same pixels the same way over long time periods. This chapter explores ways to capture and parameterize this motion.

We focus especially on capturing the motion observed by each pixel over time, because this low-level cue is applicable to many problem domains: capturing consistent motion of cars on roads, consistent motion of people in malls, and consistent motion of blood cells within blood vessels.

The first part of the chapter deals with estimating optic flow from spatio-temporal summary statistics accumulated over time. This is different from the more typical optic flow problem in which the motion of objects between a pair of consecutive images is computed. In either case, a basic tool used is the optic flow constraint, which relates the spatio-temporal image derivatives to object motion in the image. In the spatial case, spatio-temporal derivatives are collected over a local region around a pixel. However, when watching a motion pattern which remains consistent over time, we can consider the spatio-temporal image derivatives at one pixel location throughout the entire video sequence. This allows us to constrain the motion we expect in the scene based on the entire video sequence.

Our analysis is based on characterizing the spatio-temporal structure tensor, which is the covariance matrix of the spatio-temporal intensity derivatives observed at a particular pixel over time. This summary statistic can be accumulated in real time. In the context of stationary motion patterns, we characterize the failure modes of several standard methods to estimate the optic flow from the structure tensor.

While much of the first part of this chapter highlights challenges, ambiguities and failure modes of using the structure tensor to represent motion, it captures and summarizes important information about observed motions. In particular, the structure tensor allows the computation of several energy functions which can be used to score how consistent a motion vector is with observed data. Classical flow estimates including least squares, total least squares, and normal flow are solutions which minimize the error function defined on the structure tensor. However, retaining the complete information allows applications to integrate all of the information within the tensor into energy functions whose minimization defines global motion patterns.

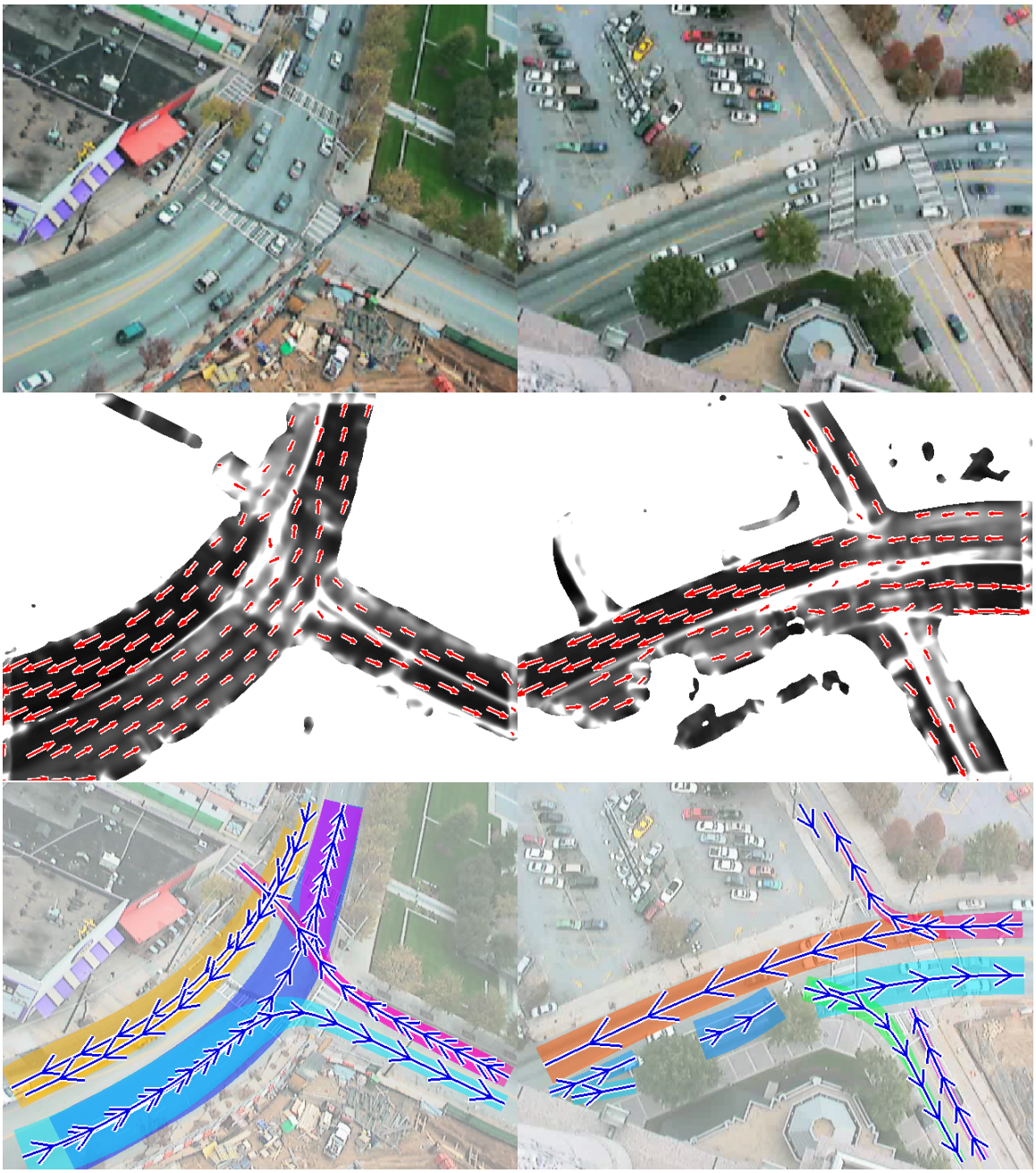


Figure 2.1: We seek to parametrize roads from video, such as the examples shown on the top. In the middle row, real-time accumulation of spatio-temporal derivatives captures both the best fitting motion direction (red arrows) and confidence values (black is more confident). On the bottom, results from a new algorithm which parameterizes roads by speed, direction and width are shown.

In this vein, in the second part of the chapter, we have created parametric snake models. Unlike standard snake models, the derivative of the snake is interpreted as a velocity and matched for consistency directly with the structure tensor using these energy functions. This produces open-ended snakes which are stable and unanchored at their endpoints. Using these snakes, we are able to automatically parameterize roads using video data of traffic from stationary or georegistered cameras. Figure 2.1 shows scenes from two video sequences from stationary cameras looking at traffic in roads. The second row in the figure shows the directions of consistent motion computed by fitting vectors to the structure tensor extracted from the video in real time. The third row shows results for the second part of this chapter in which we parameterize roads by their direction, speed, and width.

The rest of the chapter contains the following sections. Section 2.2 gives an overview of related work both in spatio-temporal derivatives and snakes. Section 2.3 gives a summary of spatio-temporal derivatives. Section 2.4 shows the spatio-temporal derivatives created by a moving object in simple scenarios. Section 2.5 develops 3 energy functions which can be used to match a velocity vector to a spatio-temporal derivative tensor. Section 2.6 analyzes the failure modes of the energy functions and shows results on both synthetic data and a traffic scene. Section 2.7 begins the second part of the chapter, in which parametric snake models are fit to the spatio-temporal derivatives, by introducing traditional cubic B-Spline snake models. Section 2.8 introduces our specialized energy functions which we use to fit snakes by velocity. Section 2.9 computes the analytical derivative of the snake energy functions. Section 2.10 presents the optimization algorithm used to fit the snake models. Finally, section 2.11 shows results of fitting snake models and explores the effects of changing scaling parameters in the algorithm.

## 2.2 Related Work

We give related work in three areas: classic optic flow and spatio-temporal derivative statistics, road delineation and snake models, and technical background specific to our use of snakes.

### 2.2.1 Optic Flow and Spatio-temporal Derivatives

Optic flow is among the most fundamental problems in computer vision [42], and for the classic problem there are methodologies and datasets for evaluation [5], fast variational solutions [14], and slow algorithms which elegantly relax standard assumptions of constant lighting or camera blur parameters [79].

Prior work focused on capturing spatio-temporal statistics at a pixel over time is targeted at motion anomaly detection [69], and motion in crowded and busy scenes [48]. There has also been work on detecting roads and motion patterns [68, 100]. Other work has captured different statistics of local motion such as distributions of optic flow vectors [61, 62], and used them for background subtraction and anomaly detection.

The use of total least squares [45] in the solution for optic flow originated as a way to reduce bias for least squares flow estimates from noisy infrared image data [93], and was also discovered to remove bias in vehicle tracking applications [64]. However, total least squares is only unbiased when the noise corrupting the $x$, $y$, and $t$-derivatives is *i.i.d.*, and there exists evidence from computational modeling of human optic illusions [30, 31] that suggest that there remains a bias in natural conditions.

### 2.2.2 Road Delineation and Snake Models

Traditional methods of road delineation are based solely on the appearance of roads. Early algorithms such as [63, 83] labeled pixels as being either roads or background by thresholding their intensities. Many improvements have been proposed to identify line segments in satellite imagery with heuristic-based line detectors [32, 98], and the Hough transform [10]. These methods work best in areas where there is minimal variation in road appearance.

Some newer approaches to road delineation are based on Active Contours, an algorithm originally introduced by Kass *et al.* [46] and popularized with the name snakes. Snakes provide a framework for segmentation which aims to balance global smoothness constraints and localized fitting to image features. Marikhu *et al.* [57] utilize higher order active contours [72] for road extraction from static satellite imagery.

However, like most previous work, their methods are based solely on road appearance and fail in the presence of occlusions and shadows.

There have been several methods for refining image based energy functions for improved snake performance. Gradient vector flow is a technique used to improve convergence of snakes to concave areas by propagating gradients to parts distant from the concave area [101]. Another technique replaces the edge energy function used in fitting a snake by a statistically based region energy function [1].

Non-parametric techniques such as level sets and parameterization free snakes have also been explored [11, 67]. These techniques are generally aimed towards segmenting closed regions, and are naturally able to accommodate topological changes. However, they are not well suited to delineating roads, which are naturally open-ended. There are, however, extensions which define the curve implicitly as the centerline of a level set, allowing the extraction of long thin structures [7].

Particular problems arise when using free-floating open-ended snakes due to the extra freedom of movement the endpoints have, generally resulting in a zero-length, trivial snake being the minimum of the energy function. In [16], networks of snakes were created and jointly optimized using the same framework as traditional snakes; however, the network endpoints were constrained to be fixed anchor points or to be along the boundary of the image. One of the advantages to the approach we give in section 2.8 is that the velocity of the snake parameterization is constrained to match the estimated velocity of the underlying motion, removing the preference for a trivial, zero length snake.

Melonakos *et al.* [59] uses Finsler Metrics to create an energy function which forces the snakes orientation to fit known orientations within the image [4]. These snakes have directional cost functions which make it much cheaper to align along a road in the direction of travel. They illustrate their methods to segment road imagery. We view our work as addressing two key limitations of this previous work. First, although Melonakos *et al.* optimize snakes which are not closed curves, these snakes have fixed endpoints. In contrast, we allow the endpoints of the snake to move freely. Additionally, the "directional data" used in [59] is not derived from motion; rather, it is artificially generated by fitting a static image template along roads. In contrast, we use the accumulation of spatio-temporal image derivatives to define a constraint

on the speed of our snakes. This gives a different kind of underlying data that more specifically exploits long term observations of motion patterns.

### 2.2.3 Technical Background Specific to Our Approach

We will consider snakes that dynamically fit to the width of a road. Fitting a snake to a structure with width can be done using multiple snakes which are connected with constraints [90, 103]. We follow a more direct approach where the thickness of the snake is directly estimated. The convergence properties of such snakes have previously been characterized in [20].

Adding width to our snakes creates a sort of motion segmentation algorithm. In general, motion segmentation starts either with optic flow computed at each frame using spatial constraints [3, 43, 54, 95] or complete trajectory data [94]. Segmentation techniques are then applied to the directional data. In [43] a vector-aware similarity measure and a hierarchical clustering algorithm is used. The Lagrangian Coherent Structures of a vector flow are found in [3] to produce a segmentation. Hierarchical Dirichlet Processes have been used both on low level motion event descriptions [95] and full trajectory data [94] to create semantic regions. In [54] event descriptors which include motion computed through optic flow are created and used to segment the scene and create behavior models using co-occurrence of events. Our method differs from motion segmentation in that it is tailored towards long thin segments (representing roads) and that it directly produces a parametric model of entire roads. Furthermore, not all motion segmentation methods are able to process sparse motion which is only seen in some frames of the video. Additionally, our method only requires summary statistics of the scene which are easy to compute and require little memory.

## 2.3 Background on Spatio-temporal Derivatives

The optic flow constraint equation [42], relates the $x, y, t$ derivatives of image intensity $(I_x, I_y, I_t)$ of an image sequence to the apparent motion of objects in the image $(u, v)$:

$$I_x u + I_y v + I_t = 0 \tag{2.1}$$

In our applications we compute the image derivatives using blurred pixel differences in the image at each pixel location. The blurring is necessary, since objects move multiple pixels between frames. The optic flow constraint equation is based on the first order Taylor series expansion which assumes the intensity change at a pixel is linearly correlated with the magnitude of the motion. Figure 2.2 shows examples of each derivative for a particular frame of the video. As written, the equation is ill-posed: spatio-temporal derivatives $(I_x, I_y, I_t)$ are estimated from image data, but there is only one equation and two unknown variables $(u, v)$. Commonly, there is an assumption that the flow vectors $(u, v)$ are constant in a region, or varying slowly over the image. Each of these assumptions can be used to provide additional constraints.

We are interested in the case where all objects which move over a pixel have similar motion, $(u, v)$ such as occurs for traffic traveling over a road or blood cells in a vascular system. This allows us to constrain equation 2.1 by collecting spatio-temporal derivatives over time instead of over space. In order to avoid including the spatial gradients of the background in the summary statistics of motion, we ignore frames where there is no object moving over the pixel and $I_t$ is close to 0. Thus, at a particular pixel, we capture image measurements as a data cloud in the form shown in figure 2.3, with the blank area in the middle of the data volume showing the band around $I_t = 0$ where we do not include data due to a lack of motion.

## 2.4 Intensity Gradients of Moving Objects

When an object moves through a scene it causes characteristic gradient changes. Consider the simple scenario of a dark object with sigmoidally blurred edges moving through an image at constant speed. In figure 2.4 we measure the changes in image

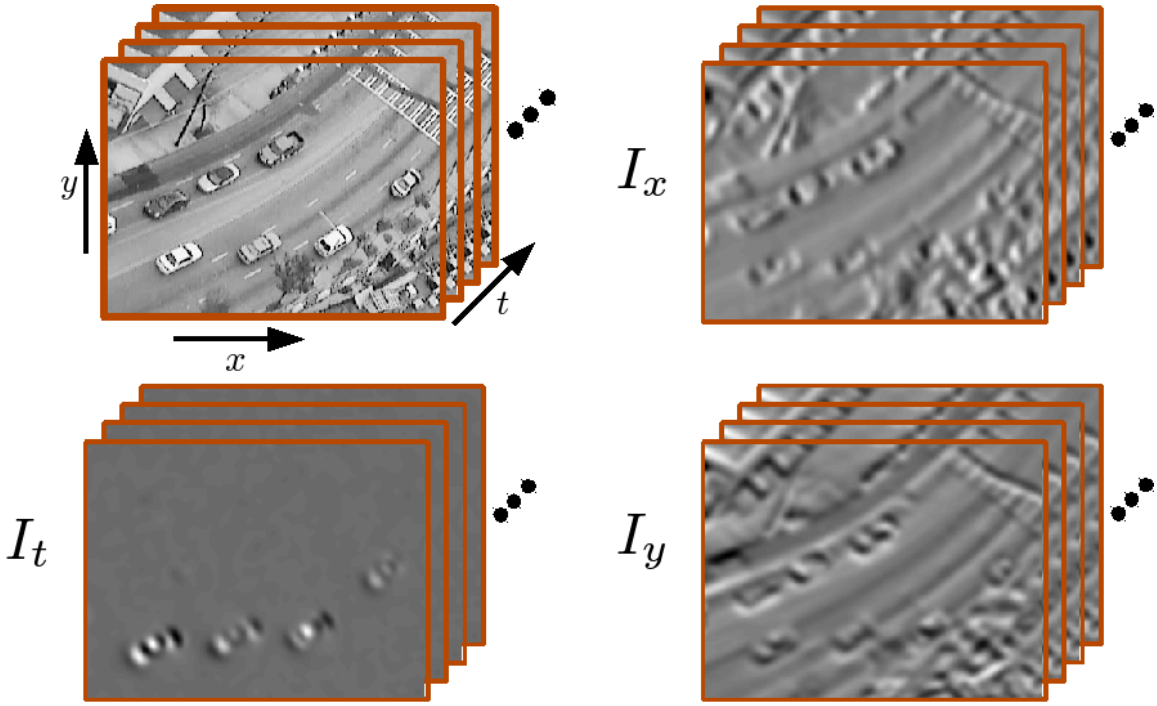



Figure 2.2: Examples of spatio-temporal derivatives computed for the image sequence shown in the upper left.

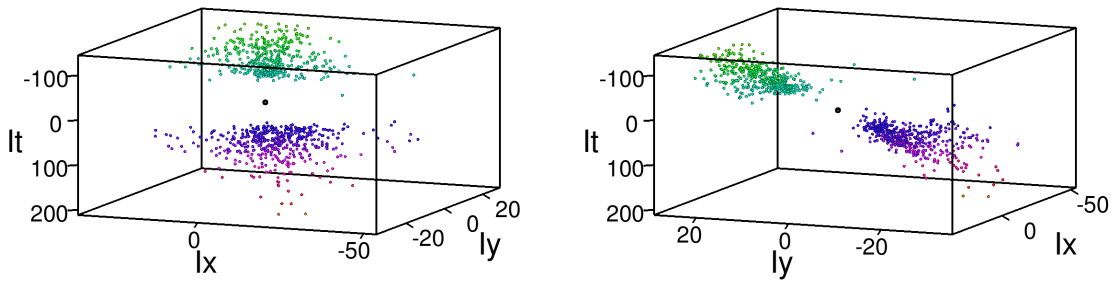



Figure 2.3: Two views of spatio-temporal derivative data, at a single pixel, displayed in the $I_x$, $I_y$, $I_t$ cube. The area around $I_t = 0$ has been filtered out, since it likely corresponds to the background state of no motion.

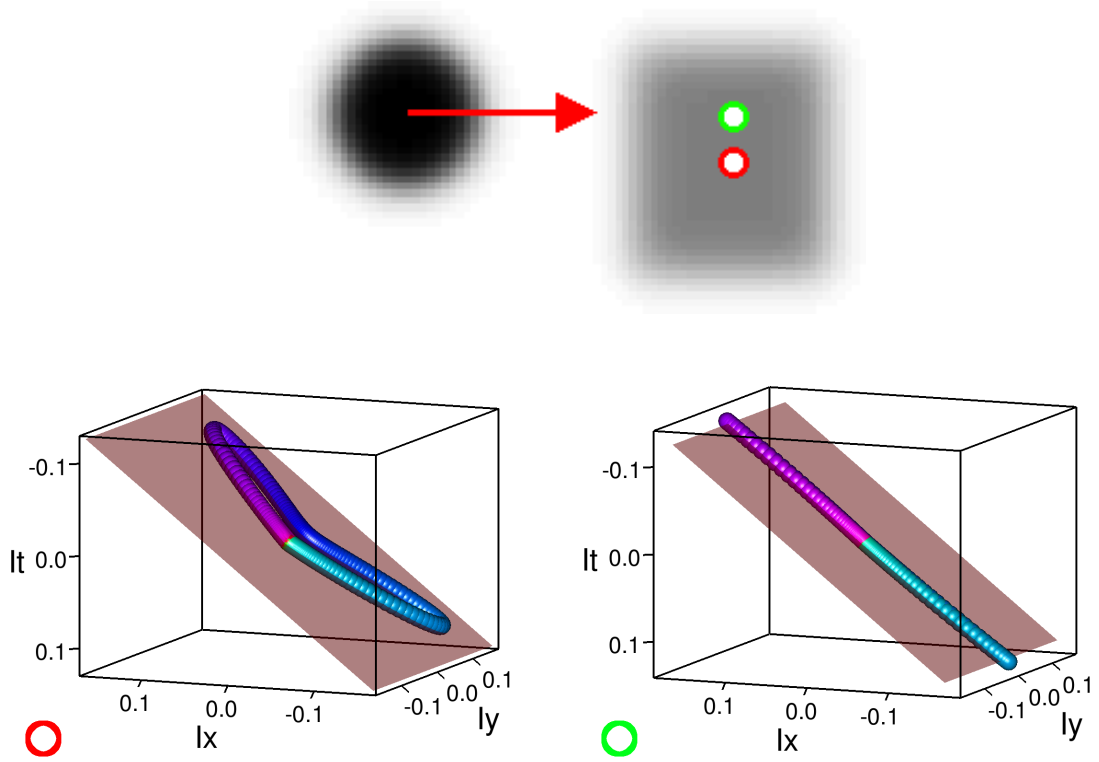



Figure 2.4: A round object is moved through a scene (top). The induced spatio-temporal derivatives are measured at two pixels, one directly in the path of the object (bottom right), the other slightly below the path of the object (bottom left). The plane drawn corresponds to the optic flow constraint and the known motion of the object.

intensity at a point over which the object traverses. The passing of the object induces a curve through spatio-temporal derivative space which starts and ends at the point $(0, 0, 0)$. The equation for the optic flow constraint, when $u$ and $v$ are fixed, define a plane of allowable values for the vector $(I_x, I_y, I_t)$.

$$I_x u + I_y v + I_t = 0 \tag{2.2}$$

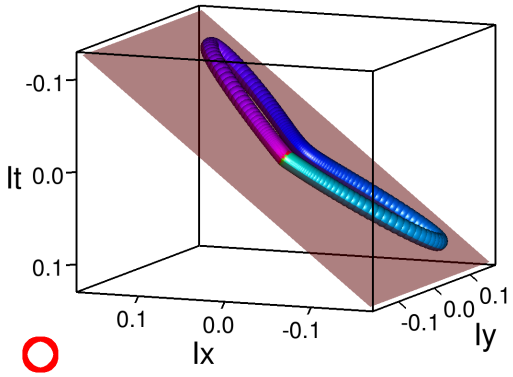
Values of $I_x, I_y, I_t$ which are consistent with the motion $(u, v)$ must reside within this plane. In figure 2.4 it can be seen that the constraint plane cuts through all the data points.

At the red pixel in figure 2.4, the observed gradients over time follow a cycle, first all three of $I_x$, $I_y$, $I_t$ are increasing; but, when the dark object passes, the observed pixel $I_x$ and $I_t$ both change signs, while $I_y$ does not. This traces out a heart shaped curve that spans the optic flow constraint plane. At the green pixel in the figure, the gradients are always completely horizontal as they pass over the point; therefore, $I_y$ is always zero and all gradients lie on a line. In this situation, any optic flow constraint plane which goes through this line will realistically explain the data. Therefore, the data might have been created by a slow moving object moving directly right or a faster moving object moving diagonally up and right (or down and right). This ambiguity is the temporal version of the aperture problem. Usually this situation only arises in the middle of a lane of traffic where the gradients are predominantly, exactly in the direction of motion. We use this fact to motivate the normal flow energy function in section 2.5.

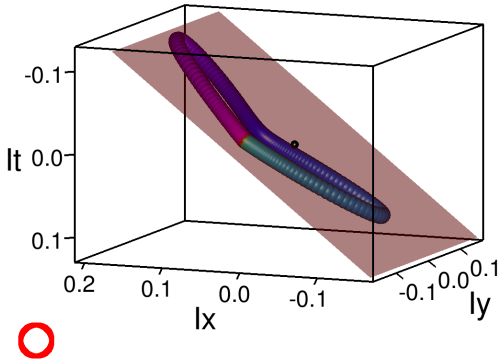
We now consider background gradients. Although unrealistic, consider the case of an object which is additive with the background (figure 2.5(a,b)).

$$(I_x - B_x)u + (I_y - B_y)v + I_t = 0 \tag{2.3}$$

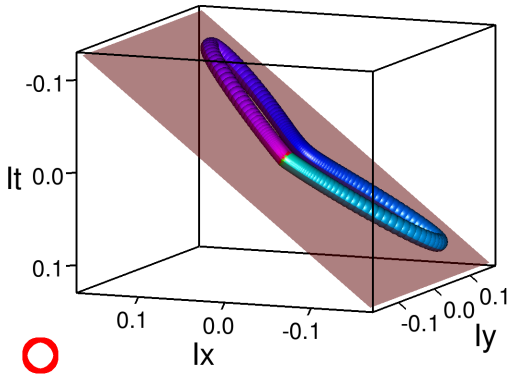
$I_x$ is the object intensity gradient and $B_x$ is the intensity gradient of the background. This equation describes a displaced plane through the point $(B_x, B_y, 0)$, the background gradient of the pixel. Now consider the more realistic scenario of an opaque object. If the object completely occludes the background, the background gradient is irrelevant and the spatio-temporal derivatives reside on the optic flow constraint



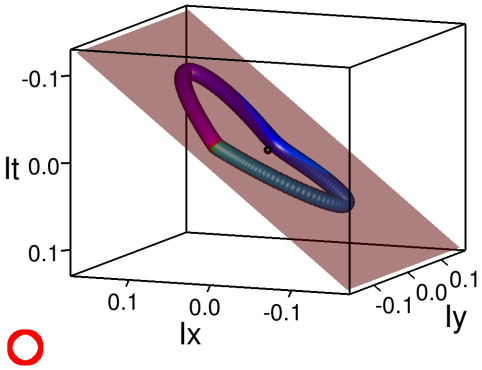

(a) Additive: No Background Gradient




(b) Additive: Background Gradient




(c) Opaque: No Background Gradient




(d) Opaque: Background Gradient

Figure 2.5: A curve is traced out in spatio-temporal derivative space by the passing of an object centered above the position of the pixel (the same distance as the object passes above the red circle in figure 2.4). (a) An additive object passes over a pixel which does not have any background gradient. (b) An additive object passes over a pixel with a background gradient in the direction of travel. Notice that the entire curve is shifted left (and is entirely under the ideal constraint plane). (c) An opaque object passes over a pixel with no background gradient, in this scenario the curve is identical to the case for an additive object. (d) An opaque object passes over a pixel with a background gradient in the direction of travel. Unlike the additive case, the spatio-temporal derivatives start at the location of the background gradient but then move into the optic flow constraint plane as the object completely occludes the background, tracing out a complicated 3 dimensional curve.

plane thourgh $(0, 0, 0)$. However, the computation of smooth image derivatives involves blurring the image features with a Gaussian. This means that when the object is far away the blurring creates something akin to an additive object, while when the object is directly over the background it is completely occluding it. In figure 2.5(d) we see the effect of an opaque object moving over a gradient. The curve starts in the displaced plane around $(B_x, B_y, 0)$ in the additive model. However, as the object moves through the image, the curve transitions into a non-displaced plane and then, when the object passes, it moves back to the displaced plane of the additive model. The object, therefore, traces out a complex three dimensional curve which is not well modeled by the optic flow constraint plane. Although the same reasoning can be applied to any background gradient, when the gradient lies in the constraint plane (because it is orthogonal to the direction of travel) it does not affect the motion estimate, since the entire curve remains within the optic flow constraint plane even when displaced. We explicitly consider the errors caused by background gradients in section 2.6.

## 2.5 Velocity Energy Functions

Using image data to solve for an optic flow vector is a long standing problem within computer vision. However, instead of making a hard decision on the optic flow at a location and then using that to infer larger scale motion patterns, it is advantageous to use the spatio-temporal structure tensor directly. To this end, we define an energy function which gives a score of how consistent a particular optic flow is with the underlying image data. Often the image data is consistent with many flow vectors, or inconsistent with all flow vectors. This flexibility in scoring flow vectors is especially important for algorithms which are working on trade-offs between smoothness of a flow field and consistency of each flow estimate with the image data.

To begin constructing an energy function, we define the complete spatio-temporal derivative matrix $M$ at a particular pixel.

$$M(x,y) = \begin{pmatrix} I_x(x,y,1) & I_y(x,y,1) & I_t(x,y,1) \\ I_x(x,y,2) & I_y(x,y,2) & I_t(x,y,2) \\ \vdots & \vdots & \vdots \end{pmatrix} \tag{2.4}$$

When constructing our energy functions, we will only require the spatio-temporal structure tensor $H$.

$$H(x,y) = M(x,y)^T M(x,y) =$$

$$\begin{pmatrix} \sum_{\forall t} I_x(x,y,t)^2 & \sum_{\forall t} I_x(x,y,t)I_y(x,y,t) & \sum_{\forall t} I_x(x,y,t)I_t(x,y,t) \\ \sum_{\forall t} I_x(x,y,t)I_y(x,y,t) & \sum_{\forall t} I_y(x,y,t)^2 & \sum_{\forall t} I_y(x,y,t)I_t(x,y,t) \\ \sum_{\forall t} I_x(x,y,t)I_t(x,y,t) & \sum_{\forall t} I_y(x,y,t)I_t(x,y,t) & \sum_{\forall t} I_t(x,y,t)^2 \end{pmatrix} \tag{2.5}$$

In this equation, $H$ and $M$ depend on the pixel location $(x, y)$ and the summations are over all frames. Note that the structure tensor $H$ is a 3 by 3 symmetric matrix at each pixel, no matter how many frames are in a video. Therefore, we only need to maintain 6 numbers per pixel, and those numbers can easily be updated as more frames of the video are observed. To visualize these matrices, consider figure 2.3: the matrix $M$ contains every point in the point cloud (as a row), while $H$ captures the covariance matrix describing the zero-mean Gaussian distribution which best models the point cloud. Figure 2.6 shows the values of the matrix $H$ for each pixel. The upper left hand image is a value of $\sum_{\forall t} I_x(x,y,t)^2$ for each pixel $(x, y)$, and so forth for all the entries in matrix $H$.

We will use $H$, the spatio-temporal structure tensor, to define three energy functions for computing how consistent a velocity is with the spatio-temporal derivative data. An optic flow velocity of best fit can be computed by finding the minimum the energy function.

We will make use of the optic flow constraint plane in constructing our energy functions. If we assume there is no background gradient and the optic flow constraint

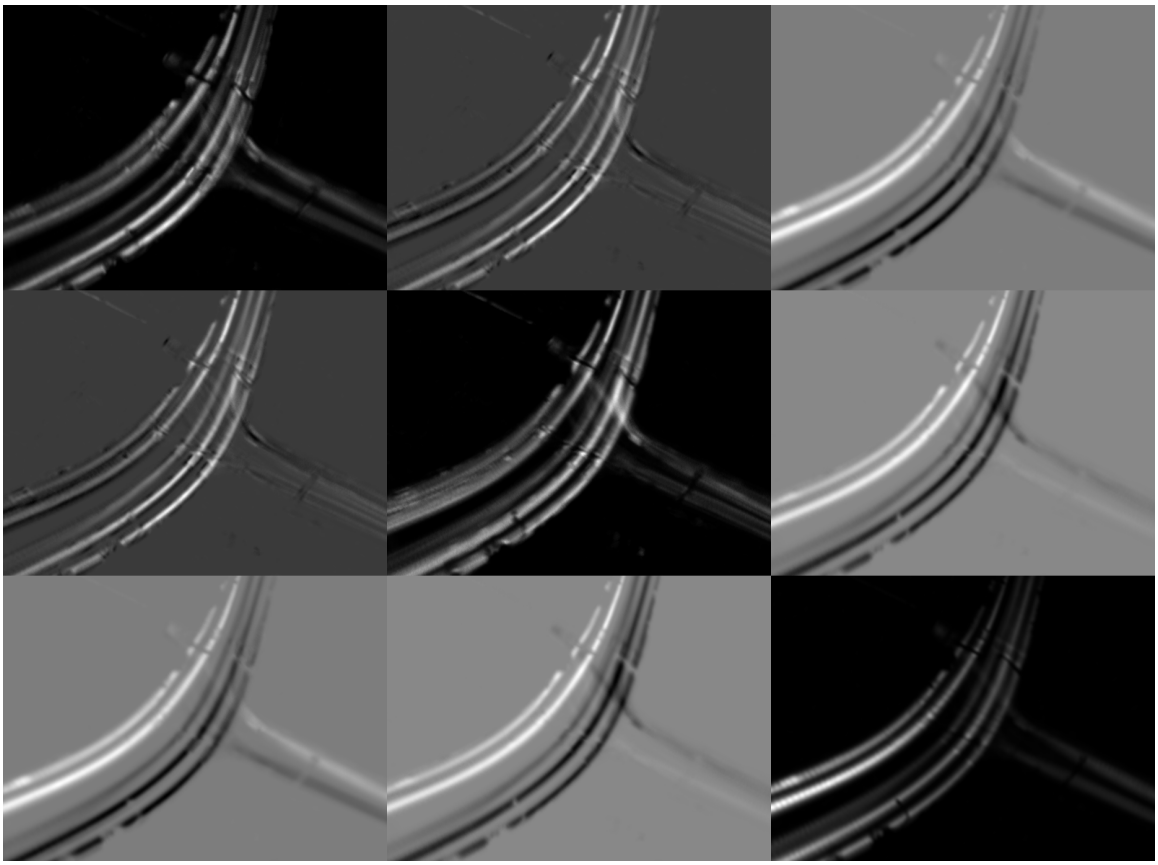


Figure 2.6: Each image shows one of the entries of the matrix $H(x, y)$ for each pixel. The upper left shows $\sum_{\forall t} I_x(x, y, t)^2$, while the upper right shows $\sum_{\forall t} I_x(x, y, t) I_t(x, y, t)$ and so forth for each entry in the matrix as shown in equation 2.5. This summary statistic captures information about how objects travel through the scene. For example, the covariance term between $x$ derivatives and $t$ derivatives shown in the upper right is different depending on which direction traffic moves over the road.

plane goes through $(0,0,0)$, then a velocity $(u,v)$ defines an optic flow constraint plane. Our first energy function will be defined as the squared $I_t$ distance between each spatio-temporal derivative and the optic flow constraint plane. The $I_t$ distance between the point $(I_x, I_y, I_t)$ is the difference between the value of the constraint plane at $(I_x, I_y)$ which is $-I_x u - I_y v$ and the value $I_t$. This quantity is $|I_x u + I_y v + I_t|$. This leads to the following cost function.

$$E_{LS} = \left\| M(x,y) \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \right\|^2 \tag{2.6}$$

This equation can be rewritten to depend only on the matrix $H$.

$$E_{LS} = \begin{pmatrix} u & v & 1 \end{pmatrix} H(x,y) \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{2.7}$$

Notice that this error function is exactly that minimized by the least squares solution of the following matrix equation.

$$\begin{pmatrix} I_x(x,y,1) & I_y(x,y,1) \\ I_x(x,y,2) & I_y(x,y,2) \\ \vdots & \vdots \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} -I_t(x,y,1) \\ -I_t(x,y,2) \\ \vdots \end{pmatrix} \tag{2.8}$$

If we assume that $I_x$ and $I_y$ have no noise and $I_t$ is corrupted with $i.i.d$ noise, then least squares is an unbiased estimator of the values $u$ and $v$.

Our second energy function will be defined as the sum of the shortest distance from each spatio-temporal derivative to the optic flow constraint plane.

$$E_{TLS} = \left\| M(x,y)\vec{g}(u,v) \right\|^2, \qquad \vec{g}(u,v) = \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} / \left\| \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \right\| \tag{2.9}$$

The vector $\vec{g}(u,v)$ is a unit vector in the direction of the normal to the constraint plane, and the matrix equation is the projection of each spatio-temporal derivative vector onto this unit vector, which is exactly the distance from the spatio-temporal

derivative to the optic flow constraint plane. The equation can be rewritten as follows.

$$E_{TLS} = \vec{g}(u,v)^T H(x,y)\vec{g}(u,v) \tag{2.10}$$

Notice that this equation contains $H$, not its inverse, as is common when minimizing a negative log likelihood of a Gaussian with covariance matrix $H$. Instead, this energy function is minimized at vectors that lie in the direction of least variation of $H$; this corresponds to the homogeneous representation of the most likely optic flow direction. An interesting property of this energy function is that its values are bounded between the largest and smallest eigenvalues of $H$ by the Reyleigh-Ritz theorem.

This energy function $E_{TLS}$ is minimized at the total least squares solution of the following matrix equation.

$$M(x,y)\begin{pmatrix} wu \\ wv \\ w \end{pmatrix} = \vec{0}, \qquad \left\| \begin{pmatrix} wu \\ wv \\ w \end{pmatrix} \right\| = 1 \tag{2.11}$$

The total least squares solution is an unbiased estimator when the entries of the matrix $M$ have been sampled with *i.i.d.* noise.

Since $I_x$, $I_y$ and $I_t$ are all estimated from the same images, all three will have noise in them. This suggests that the total least squares approach should be used, since it makes the most reasonable assumption about the distribution of noise. However, notice that $I_x$ and $I_y$ are measured in brightness change per pixel, while $I_t$ is measured in brightness change per frame. There is no clear way of converting the scale from pixels to frames. Additionally, images are blurred spatially, but often not blurred temporally, so spatial derivatives may have smaller noise than temporal derivatives and in any case are likely to have different noise distributions (not *i.i.d.* noise). The smaller noise in the spatial dimensions give good motivation for using the least squares solution rather than the total least squares solution.

As a thought experiment consider changing the arbitrary scaling between pixels and frames. We can introduce a scaling factor to each spatio-temporal derivative and rescale our output velocities accordingly. Surprisingly, the least squares solution gives the same optimal velocity, after rescaling, regardless of the scaling factor. The

total least squares solution, on the other hand, changes to reflect the modified relative importance of temporal and spatial errors. Furthermore, as the scaling factor goes to zero, the $I_t$ direction collapses to nothing, and the rescaled total least squares and least squares solution become the same. This provides additional motivation for using the least squares solution, since it is simpler and does not depend on a scaling factor between pixels and frames. Additionally, we will see in section 2.6 that the total least squares solution suffers from errors in common situations in which least squares is unaffected.

Our third energy function will be based on fitting a line to our data instead of a plane. If we assume that the observed spatial gradient is always directly in the direction of travel, then the spatio-temporal derivatives will lie along a single line in the $I_x, I_y, I_t$ space. We create an energy function, given a velocity vector $(u, v)$, by finding the constraint line and then seeing how well the spatio-temporal derivative data matches this line. We will call this line the primary gradient line and parameterize it by the unit vector $(L_x, L_y, L_t)$. The following equations hold, since this line must lie in the optic flow constraint plane and the direction of travel $(u, v)$ is in the same direction as the gradient $(L_x, L_y)$.

$$\begin{aligned} &L_x u + L_y v + L_t = 0 && \text{(Optic flow constraint plane)} \\ &L_x^2 + L_y^2 + L_t^2 = 1 && \text{(Unit vector)} \\ &u = kL_x, \quad v = kL_y && \text{(Gradient in direction of movement)} \end{aligned} \tag{2.12}$$

The scaling factor $k$ is arbitrary and must be solved for. We have a system of 4 equations with two given variables $u$ and $v$ and 4 unknown variables $L_x$, $L_y$, $L_t$, and $k$. This system can be solved for the primary gradient line.

$$\begin{aligned} L_x &= \frac{-u}{\sqrt{(u^2+v^2)(u^2+v^2+1)}} \\ L_y &= \frac{-v}{\sqrt{(u^2+v^2)(u^2+v^2+1)}} \\ L_t &= \frac{\sqrt{u^2+v^2}}{\sqrt{u^2+v^2+1}} \end{aligned} \tag{2.13}$$

Our error function will be based on the length of the projected distance of the spatio-temporal derivative vector onto the primary gradient line.

$$\left\| M(x,y) \begin{pmatrix} L_x \\ L_y \\ L_t \end{pmatrix} \right\|^2 = (L_x \quad L_y \quad L_t)\ H(x,y) \begin{pmatrix} L_x \\ L_y \\ L_t \end{pmatrix} \tag{2.14}$$

The higher this number is, the better the spatio-temporal derivative vector is modeled by the primary gradient line. To create an energy function which we will minimize, we take the negative of this quantity and, for convenience, add the trace of the matrix $H$ to ensure we always have a positive value.

$$E_{normal-flow} = tr(H) - (L_x \quad L_y \quad L_t)\ H(x,y) \begin{pmatrix} L_x \\ L_y \\ L_t \end{pmatrix} \tag{2.15}$$

The trace of $H$ equals the sum of the eigenvalues of $H$ all of which must be non-negative since $H$ is a covariance matrix (positive-semidefinite). Therefore, this quantity will always be larger than the quadratic term on the right, which is bounded to be smaller than the largest eigenvalue of $H$ by the Reyleigh-Ritz theorem. The optimal velocity of best fit, for this energy function, will be the one corresponding to a primary gradient line $(L_x, L_y, L_t)$ in the direction of the eigenvector corresponding to the largest eigenvalue of the covariance matrix $H$. As with $E_{TLS}$, this energy functions is bounded within the range of the eigenvalues of the matrix $H$.

The three energy functions presented in this section represent different ways of determining how consistent a velocity vector is with the spatio-temporal derivatives seen at a pixel. They can be used to compute velocity fields of best fit. However, they can also be used directly within an application, as will be seen later when fitting snakes to spatio-temporal derivative data.

Before continuing with an analysis of the energy functions, we need to address the collection of the spatio-temporal derivative statistics. Note that each pixel has its own $M(x,y)$ and $H(x,y)$ matrices. For each pixel, we do not maintain the complete matrix $M(x,y)$ but only the covariance summary $H(x,y)$. For stability, $H(x,y)$ is only updated for frames in which significant temporal variation is observed at the

pixel location. A count of the number of frames used is maintained in a variable $C(x, y)$. We ignore all pixels which have too small a $C(x, y)$ value, since they will be inherently unreliable due to a lack of data.

## 2.6 Analyzing Object Motion

In this section, we analyze the three energy functions using a synthetic example, a traffic scene, and video of blood flow in a Zebrafish. In figure 2.7, three lanes of synthetic traffic are shown in which we send small circular objects at various speeds. In the top lane, objects move one after the other, slowly from left to right. In the middle lane, objects move quickly and in the bottom lane, objects of various speeds are seen. To illustrate the dynamics of an intersection we have sent objects of various speeds upward in the middle of the image. The least squares solution vectors are shown in red, the total least squares solution vectors are shown in green, and the optimal velocity vectors for the $E_{normal-flow}$ energy function are shown in blue. In general, the least squares solution produces the best vectors.

In the top lane, all methods give accurate results on a flat background and when the background gradient is orthogonal to the direction of travel. In the intersection, there are small differences in the results from the three methods; however, all of the methods point diagonally between the two directions of travel observed. When the gradient is in the direction of travel, a bias is introduced due to a shifting of the rest state and object trajectories which do not lie on a simple plane, as was discussed in section 2.4. In this region the normal flow method is biased to point more towards the center of the lane, whereas both least squares and total least squares are biased to point towards the outside of the lane of traffic. This is because least squares and total least squares tilt the secondary axis of the optic flow constraint plane to move closer to the shifted spatio-temporal derivatives (see figure 2.5(d)). However, the normal flow method tilts the primary axis of the optic flow constraint plane closer to the shifted points, causing the estimated direction to be biased in the opposite direction.

In the center lane, the same observations as in the top lane can be made, however, the effects are more pronounced. Additionally, on the right side, where the background gradient is in the direction of travel, there are differences between the total

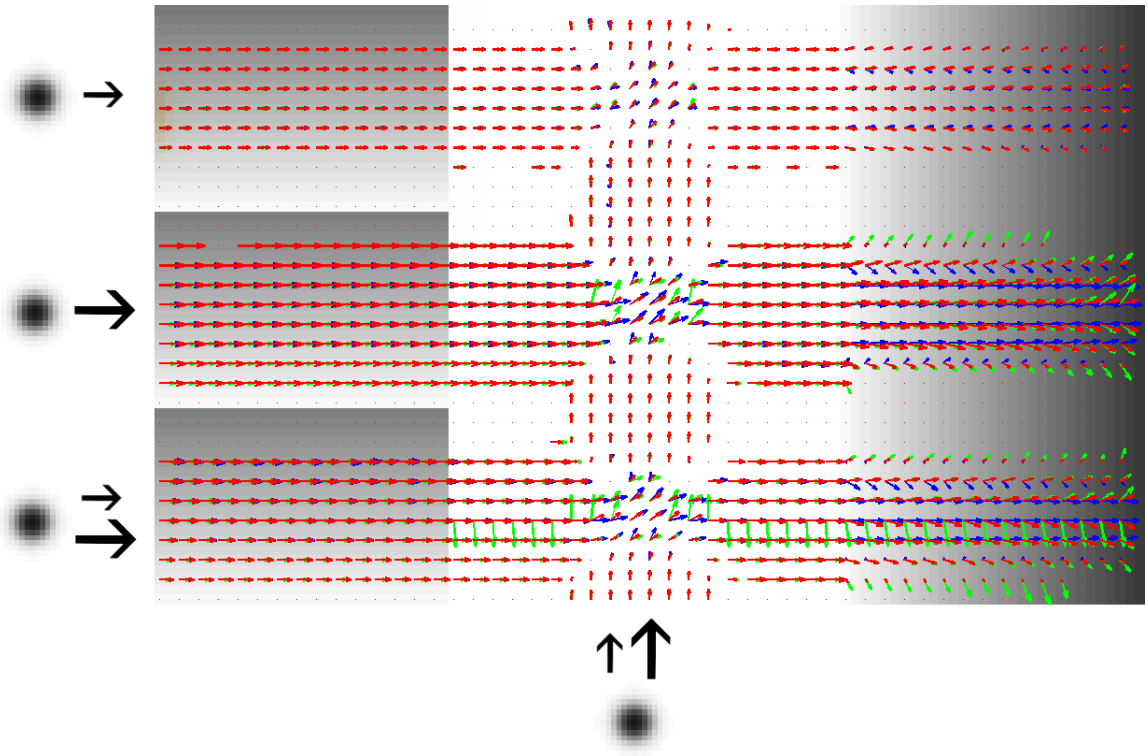


Figure 2.7: We test our methods by moving a succession of objects across an image with some background gradients and estimate the direction of best fit by minimizing the three energy functions. Red is least squares, green is total least squares, and blue is normal flow estimation. The top has slow moving objects, the center fast moving objects, and the bottom both slow and fast objects. In the middle of the image, we send objects upward. On the left, we have background gradients which are orthogonal to the direction of travel and do not generally pose a problem to our methods. On the right, we have background gradients in the direction of travel, which bias our results. In the bottom row, we see a case where, in the middle of the lane, total least squares completely misestimates the optic flow direction due to the aperture problem and different speed objects.

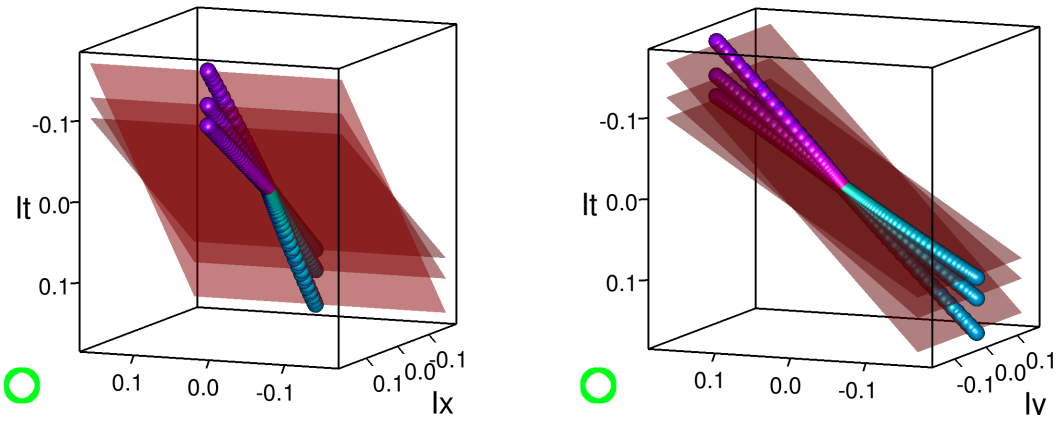



Figure 2.8: Two views of the intensity gradient space as three objects with different speeds pass directly over the pixel (see the green circle in figure 2.4). This configuration poses a problem for the total least squares method since the optimal way to fit a plane through these points is vertically and this plane corresponds to a very large velocity nearly orthogonal to the true optic flow.

least squares method and the least squares method. At the edges of the lane, the total least squares method produces completely inaccurate vectors because it fits the optic flow constraint plane near vertically to the biased spatio-temporal derivatives. Least squares, on the other hand, is unable to fit an optic flow constraint plane near vertically, since this produces large errors in the $I_t$ direction.

In the bottom lane, both fast and slow objects are observed. The total least squares solution produces surprising results in the center of the lane when there is no background gradient. At these locations only gradients in the direction of travel of the object are seen. As shown in figure 2.8, the passing of each object forms a line in the spatio-temporal derivative space, with slope corresponding to the speed of the object. The planes shown in this figure correspond to the optic flow constraint plane for each of the passing objects. Since the total least squares solution measures errors directly to the fit optic flow constraint plane, it places the plane vertically instead of at one of the true constraint planes, producing a large output vector that is nearly orthogonal to the true direction of travel. The least squares solution does not fall in this trap since it cannot fit the optic flow constraint plane vertically since this would produce large errors for some points in the $I_t$ direction. It is important to note, that while the total least squares solution has an optimal value which is completely wrong, in

these cases, the true velocity of any of the passing objects still has a relatively low $E_{TLS}$ energy value compared to vectors which are not consistent with the observed spatio-temporal derivatives.

We now turn our attention to the road scene in figure 2.9. The optic flow direction of best fit for the three estimation methods are displayed as colored arrows. For this dataset, least squares (red) gives the best and most consistent results. The normal flow method (blue) has many instances where it points diagonally off the road or lane of traffic, but is accurate in the middle of traffic lanes where the aperture problem is present. Total least squares (green) generally gives vectors very similar to least squares. However, because of noise, when the true velocity is small, it sometimes produces vectors that are very large and oriented in an arbitrary direction.

We highlight other interesting features of the road scene by visualizing the 3 eigenvalues of the structure tensor in false color (figure 2.10). The amount of red corresponds to the ratio of the smallest eigenvalue to the middle eigenvalue, the amount of green corresponds to the ratio of the middle eigenvalue to the largest eigenvalue, and the amount of blue corresponds to the ratio of the smallest eigenvalue to the largest eigenvalue. Because the eigenvalues are all positive, all these ratios are between 0 and 1. When the structure tensor clearly describes a plane corresponding to the optic flow constraint plane, then the first two eigenvalues are large and the third eigenvalue is small, this produces the color green or black. When the structure tensor describes a line because of the aperture problem, then the first eigenvalue is large and the second and third are both small producing the color red. In problematic regions, where the structure tensor does not describe a plane or a line, the color produced is cyan, purple, or white. In the center of the roads, an overhead view along the center of a lane sees mostly edges (such as the front and rear bumper) whose gradients are oriented exactly in the direction of motion of the car. This causes the temporal form of the aperture problem and shows up in red.

Another example application is the estimation of blood flow in Zebrafish, shown in figure 2.11. Zebrafish are translucent animals used experimentally because their circulation is visible in video microscopy. Individual blood cells are visible in the video sequence; however, they are crowded, overlapping, and partially occluded by visible anatomical structures. The zoomed in circle shows the optic flow based on a

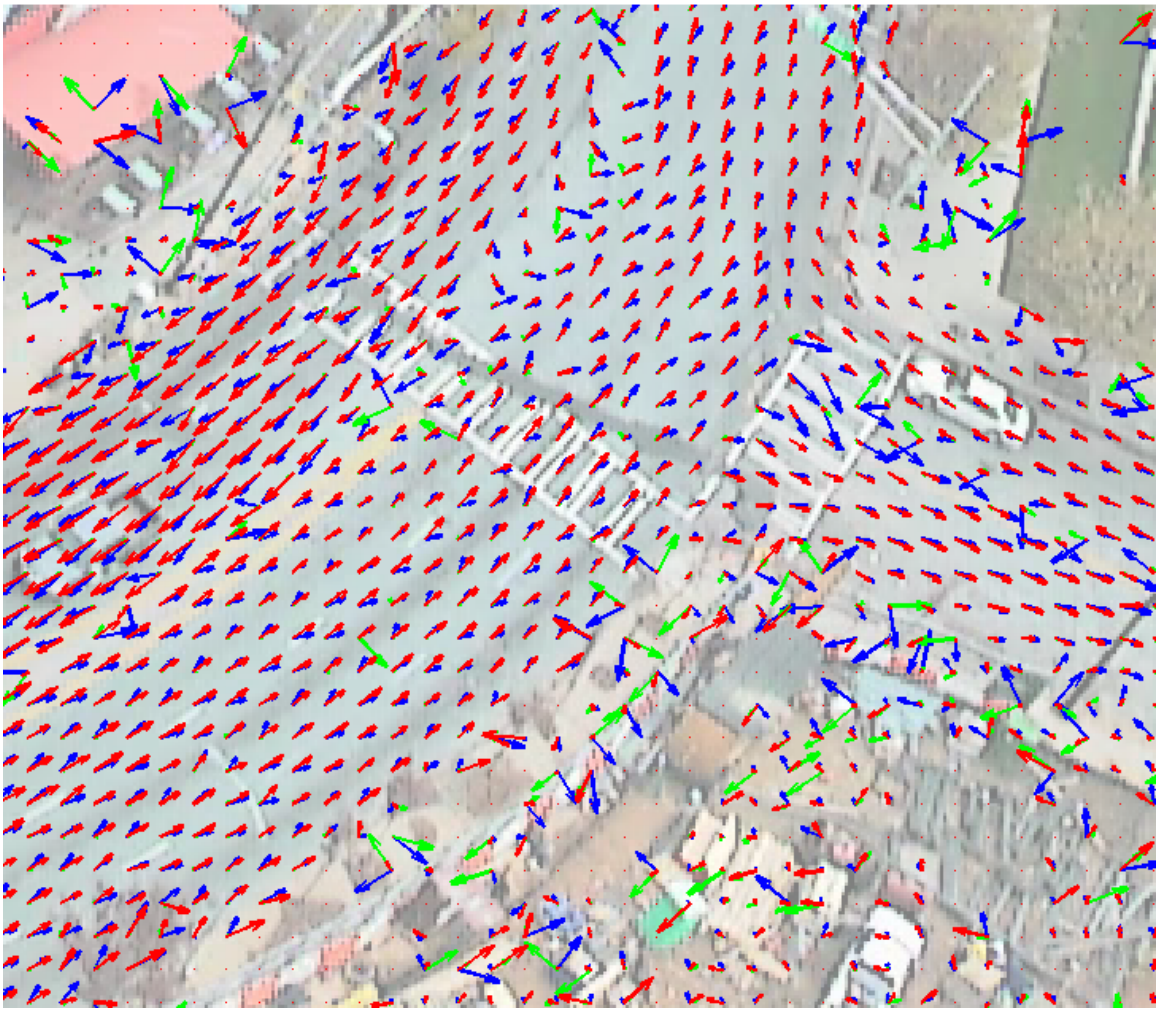


Figure 2.9: Our three methods for computing optic flow from spatio-temporal derivatives are displayed for a traffic video. In red, the least squares solution, which is best in most circumstances. In green, the total least squares solution which is less stable due to the aperture problem. In blue, the normal flow method which assumes that gradients are always directly in the direction of travel. For this data set, least squares is the best and most consistent method. The normal flow method has many instances where it points diagonally off the road or lane of traffic.

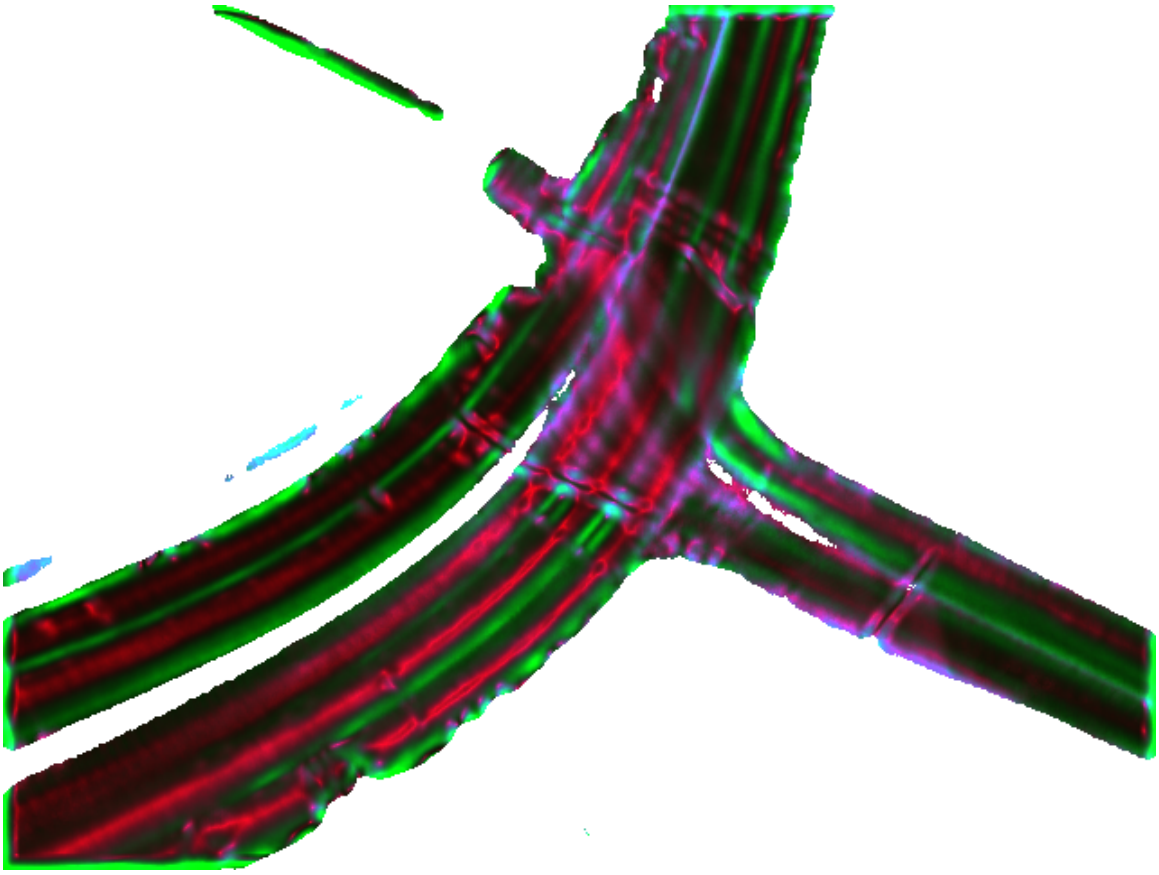


Figure 2.10: The 3 eigenvalues of the matrix $H$ for each pixel are drawn in false color with each color representing a ratio between two eigenvalues. Red is the ratio of the smallest eigenvalue to the middle eigenvalue. Green is the ratio of the middle eigenvalue to the largest eigenvalue. Blue is the ratio of the smallest eigenvalue to the largest eigenvalue. Areas in which we are confident about our estimate, the first two eigenvalues are large, and the third small, causing the image to be green or black. Areas in which the aperture problem is significant, the first eigenvalue is large in relation to both the second and the third and the image is red. Areas in which we are not confident are purple, cyan, or white.

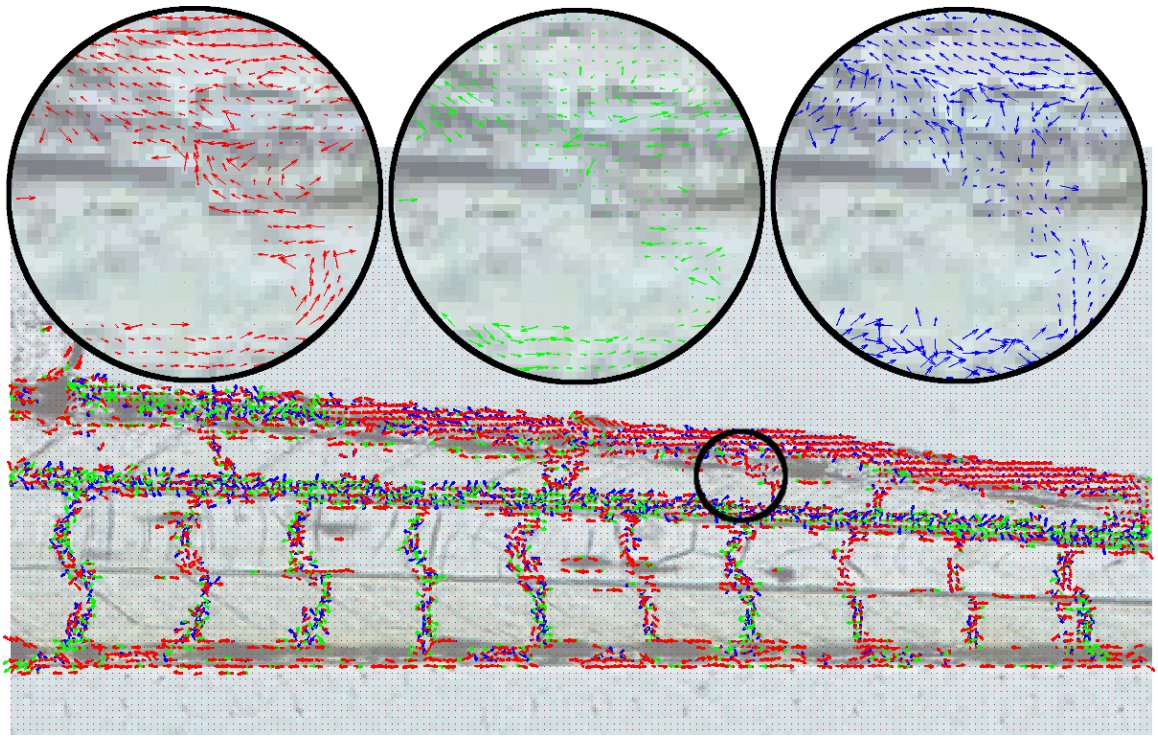


Figure 2.11: Motion patterns captured from video-microscopy of a larval Zebrafish. Capturing statistics of the spatio-temporal derivative through time gives automated ways of estimating flow patterns even when visible blood cells are crowded and challenging to track. Each criteria is shown in a different color: the least squares solution (red) gives more consistent results than the total least squares solution (green) and the normal flow estimate solution (blue). Unlike in an optic flow setting between two frames of a video, where total least squares is generally used, when considering an entire video sequence, the least squares solution gives the best results.

least squares estimator (red), a total least squares estimator (green) and a normal flow estimator (blue). The cluttered background creates strong background gradients, which introduce bias into the motion estimates. Additionally, the highlighted circle of data has relatively little movement within it and not all movement is along a consistent path. Of the three methods, total least squares has the most difficulty with this challenging example, producing many vectors of very small magnitude and several bad estimates pointing in a completely incorrect direction. The least squares estimates are generally correct and consistent. Normal flow for this example is relatively accurate in the top portion of the circle, but inaccurate in the center and bottom portions. The inaccuracies in the estimation method are caused by three factors: the strong background gradients, the inconsistent movement of objects, and the limited number of cells which move through the scene.

## 2.7 Parametric Snake Models

In the second part of the chapter we model roads using a parametric snake model. Snakes are a method of finding a parameterized curve $\vec{f}(s) = (f_x(s), f_y(s))$ which minimizes some energy function over its length. Traditionally, the snake curve is a closed loop which is expanded or contracted around an area of interest. There are usually two types of energy functions which are minimized. External energy functions match the snake to some external data, such as edges, lines, or other features in an image. Internal energy functions enforce properties of the curve itself, such as maintaining a smooth curve, or growing (or shrinking) a curve.

We use cubic B-Splines as our parametric curve due to their simplicity and differentiability. These curves are piecewise, smooth cubic polynomials controlled by a small number of basis weights (frequently thought of as control points).

$$\vec{f}(s) = \sum_{a=1}^{4} A_a(s)\vec{w}_a \tag{2.16}$$

$A_a$ are basis functions with weights $\vec{w}_a$. Due to the finite support of the basis functions, there are exactly 4 non-zero basis functions at any parameter value $s$.

Figure 2.12: An image of $R(x, y)$, a traditional style energy image which highlights features of interest, in this case, the roads in the scene. A snake can be fit to this data by aligning it along the (black) areas of low image value corresponding to an energy function which is low along lanes of traffic.

### 2.7.1 Traditional Snake Energy Functions

This section gives an overview of traditional energy functions used in snakes. The main external energy functions in traditional snakes minimizes the image energy $R(x, y)$ observed over the snake.

$$E_{image}(s) = R\Big(\vec{f}(s)\Big) \tag{2.17}$$

Common energy functions for $R$ include measures that are low in regions of edges, encouraging snakes to follow contours on an image. Figure 2.12 shows the energy function proportional to $-C(x, y)$, defined in section 2.5, highlighting areas with common intensity changes as low energy black regions.

There are traditionally two internal energy functions. The first encourages the snake to be smooth by penalizing its curvature and the second encourages the snake to grow by penalizing the negative of the magnitude of the first derivative.

$$E_{smooth}(s) = \left\|\vec{f}''(s)\right\|^2 \tag{2.18}$$

$$E_{growth}(s) = -\left\|\vec{f}'(s)\right\|^2 \tag{2.19}$$

The complete snake energy function is a linear combination of these energy terms with weights $\alpha, \beta, \gamma$ which must be empirically determined. The energy function is integrated over the entire length of the snake to create an energy function which is minimized to produce the best snake model.

$$\hat{E}_{traditional} = \alpha \int E_{image}(s)\ ds + \beta \int E_{smooth}(s)\ ds + \gamma \int E_{growth}(s)\ ds \qquad (2.20)$$

## 2.8 Velocity Snake Energy Functions

In this section, we explore several different energy functions for fitting snake models to roads using spatio-temporal derivative data. We include several energy functions to avoid problems specific to our velocity constrained snakes. Additionally, we add energy functions to allow the snake to have a dynamically adjustable width, so as to cover the entire road.

We make a fundamental change from the traditional snake model by proposing an external energy function which depends on the derivative of the snake instead of its position. The snake derivative is considered as an object velocity and fit to spatio-temporal derivative structure tensors. A major advantage obtained by using the derivative of the snake is that the snake can be an open curve with stable endpoints which do not have their position explicitly fixed. This is not generally possible with traditional snakes. Constraining the derivative of the snake instead of its position also has two minor side effects. First, there is the possibility that the snake drifts along or off the road. Second, the snake can slow to a stop at a particular position. We introduce two new energy functions to combat these effects.

Of the three traditional energy functions presented in the last section, we will only use $E_{smooth}$. $E_{image}$ will be replaced with an energy function $E_{external}$ which relates the derivative of the snake with the spatio-temporal derivatives and $E_{growth}$ is not needed since our snakes are not closed loops.

### 2.8.1 Fitting Snakes by Velocity

We fit snakes to the spatio-temporal derivative structure tensor by minimizing the energy functions developed in section 2.5. This interprets the derivative of the snake as a velocity and matches it to the spatio-temporal structure tensor through the use of one of our previously described energy functions.

$$E_{external} = \int E(\vec{f}'(s))ds \tag{2.21}$$

The energy function used can be any of $E_{LS}$, $E_{TLS}$, $E_{normal-flow}$. The use of this method with $E_{TLS}$ was first published by us in [36].

Although we showed the least squares energy function gives the most sensible optic flow vectors, the total least squares energy function is a better method for fitting snakes to spatio-temporal structure tensors. The main advantage of the total least squares approach is that the energy function is bounded by the Reyleigh-Ritz theorem to be within the smallest and largest eigenvalues of $H$. Furthermore, we scale $H$ so its largest eigenvalue has a value of 1, simplifying the use of the $E_{TLS}$ energy function. The least squares approach, since it does not normalize the velocity vector, provides no bounds. In the context of a snake, having a bounded energy function produces much better behaved snakes with consistent error magnitude across their entire lengths.

The advantage of using the spatio-temporal structure tensor as opposed to an optic flow estimate is that it includes information about which alternate directions are nearly as good as the optimal direction. For example, in an intersection, there is inherent ambiguity in the structure tensor which is ignored if only the best fit flow vector is used.

Since our snake is defined over the continuous domain but each $H$ is only defined discretely at a pixel, we use bilinear interpolation of the $H(x, y)$ matrices surrounding the point $\vec{f}(s)$ to compute equation 2.21. Computing the analytic derivative of this energy function with respect to the B-Spline control points, taking into account both the bilinear interpolation and the homogeneous coordinates is described in section 2.9.2.

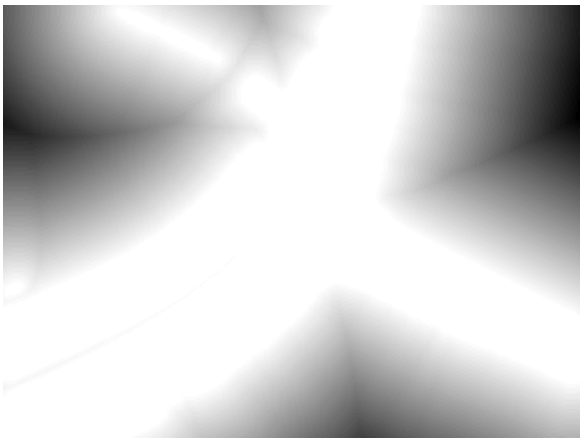


Figure 2.13: An image of $N(x, y)$ used in the computation of $E_{off-road}$. This image is zero (white) on the road. For pixels that are not on the road the distance from the pixel to the closest point on the road is used as the value.

### 2.8.2 Fitting Snakes To Vector Fields

Although we have focused on using the energy functions developed earlier in the chapter ($E_{LS}$, $E_{TLS}$, $E_{normal-flow}$) directly, we can also first compute a vector field and then match the velocity of the snake to this vector field. The vector field can be computed using the best fit vectors created by minimizing any of the three energy functions. Once we have computed the vector field we can fit the snake to it using the following energy function.

$$E_{velocity}(s) = \left\| \vec{f}'(s) - \vec{V}(\vec{f}(s)) \right\| \tag{2.22}$$

This energy function directly compares the derivative of the snake and the vector field. The analytically computed derivative of this energy function is given in section 2.9.3.

### 2.8.3 Drift Prevention

Since the optic flow equation only constrains the derivative of the snake, small errors in the spatio-temporal derivative can cause the snake to drift along the road, both forwards and sideways. We define an energy function which prevents the snake from drifting off the road by penalizing locations which are completely off the road. Recall

that we maintain a count, for each pixel, of the number of frames with significant image variation $C(x, y)$. Pixels with a high $C(x, y)$ have seen a lot of motion and are usually near the road. We use a threshold $\gamma$ to define a function $N(x, y)$ which increases with distance from a road.

$$N(x,y) = \begin{cases} 0 & \text{if } C(x,y) \geq \gamma \\ d & \text{otherwise} \end{cases} \tag{2.23}$$

In this equation $d$ is the distance to the closest pixel with sufficient image data. This score creates useful gradients in locations far from regions of interest.

$$E_{off-road}(s) = N\left(\vec{f}(s)\right) \tag{2.24}$$

Figure 2.13 shows an example image of $N(x, y)$.

### 2.8.4 Minimum Speed

It is possible for the snake to come to a stand still at a location where $H$ specifies a small speed. To prevent this, we add an energy function that is large when the snake is moving too slowly.

$$E_{minimum-speed}(s) = \left\| \frac{\vec{f}'(s)}{\alpha} \right\|^{-\beta} \tag{2.25}$$

The number $\alpha$ depends on the minimum speed we wish to maintain, and $\beta > 1$ determines the harshness of the penalty we impose on small speeds. For our experiments, we chose $\alpha = 1\,\textit{pixel per frame}$ and $\beta = 3$. The energy function should be very close to zero under normal operations; however, when the snake slows down drastically (with our parameter choices 10 times slower than the target value) then this energy term becomes large and forces the snake to move over the region at a minimum speed.

### 2.8.5 Snake Thickness

From ground-based video cameras, roads often have significant thickness. We model this thickness within the snake framework by expanding the snake in the direction

perpendicular to the tangent of the curve. For a given energy function $E$, comprising any of the energy functions considered so far, the overall energy which was formerly computed as:

$$\int E(\vec{f}(s))ds \tag{2.26}$$

now becomes an area integral, going both along the snake and, at each location $s$, perpendicular to the snake with a width $2f_\theta(s)$:

$$\int_{s\in[0,1]} \int_{p\in[-1,1]} E(\vec{f}(s) + pf_\theta(s)\tilde{\vec{f}}'(s)^\perp)\ dp\ ds \tag{2.27}$$

where $\tilde{\vec{f}}'(s)^\perp$ is a unit vector in the direction perpendicular to the tangent of the snake at location $\vec{f}(s)$. We parameterize the thickness along the snake $f_\theta(s)$ as an additional B-Spline.

In general, we would like the snake to fill the entire road that is traveling in a particular direction, thus we include an energy function which forces the snake to grow in thickness:

$$E_{thicker} = -\int f_\theta(s)ds \tag{2.28}$$

Minimizing this function simply encourages the snake to grow as large as possible.

Additionally, we want the snake to be a consistent width, so we penalize changes in thickness by penalizing the squared first derivative of the width.

$$E_{change-thick} = \int f'_\theta(s)^2 ds \tag{2.29}$$

When adding thickness, we first find and parameterize the road using a fixed width snake. Then a second round of optimization is performed which allows the width of the road to be adjusted dynamically. We found this two step approach to be necessary to avoid the situation where the ends of the snake collapse to nothing in order to overfit a small area of the image.

## 2.9 Derivative of Energy Function

In this section, we demonstrate the ability to analytically compute the derivative of the energy functions with respect to B-Spline control points. This allows fast optimization tools to solve for the control points which minimize the overall energy. All of the energy functions we have presented are differentiable, we show the analytic derivatives for the two most difficult energy functions: $E_{TLS}$ and $E_{velocity}$.

### 2.9.1 Derivatives of the B-Spline

Recall that cubic B-Splines are weighted sums of basis functions exactly 4 of which are non-zero at any parameter value $s$.

$$\vec{f}(s) = \sum_{a=1}^{4} A_a(s)\vec{w}_a \tag{2.30}$$

To minimize the energy, we must compute the derivative of the energy with respect to the basis function weights $\vec{w}$. These weights are sometimes thought of as control points; however, the B-Spline does not generally go through these points. Each weight variable is associated to one of the dimensions of the B-Spline, either the $x$ or $y$ dimensions.

$$\vec{f}(s) = \begin{pmatrix} f_x(s) \\ f_y(s) \end{pmatrix}, \quad \vec{w} = \begin{pmatrix} w_1^{[x]} & w_2^{[x]} & \cdots & w_n^{[x]} \\ w_1^{[y]} & w_2^{[y]} & \cdots & w_n^{[y]} \end{pmatrix} \tag{2.31}$$

The following derivatives will be needed to compute derivatives of energy functions. The derivatives of the B-Spline with respect to its parameterization $s$ is a weighted sum of the derivatives of the basis functions.

$$f'_x(s) = \frac{df_x(s)}{ds} = \frac{d}{ds}\sum_{a=1}^{4} A_a(s)w_a^{[x]} = \sum_{a=1}^{4} A'_a(s)w_a^{[x]} \tag{2.32}$$

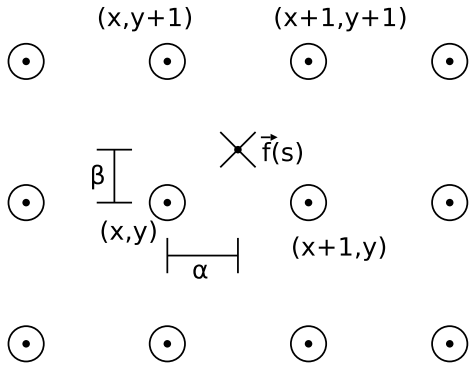



Figure 2.14: An illustration of bilinear interpolation. The matrix $H$ is only defined at each of the pixels. The location ($x$,$y$) is the nearest pixel with coordinates smaller than $\vec{f}(s)$. The variables $\alpha$ and $\beta$ are defined as the fractional parts of the vector $\vec{f}(s)$ in the $x$ and $y$ direction respectively.

The derivative of the B-Spline with respect to one of its basis function weights simply picks out the corresponding basis function.

$$\frac{df_x(s)}{dw_\tau^{[x]}} = \frac{d}{dw_\tau^{[x]}} \sum_{a=1}^{4} A_a(s) w_a^{[x]} = A_\tau(s) \tag{2.33}$$

The mixed second derivative of the B-Spline with respect to its parameter $s$ and a basis function weight $w_\tau^{[x]}$ is similarly computed.

$$\frac{df'_x(s)}{dw_\tau^{[x]}} = \frac{d}{dw_\tau^{[x]}} \sum_{a=1}^{4} A'_a(s) w_a^{[x]} = A'_\tau(s) \tag{2.34}$$

A derivative of the x-axis spline with respect to a y-axis basis function weight is zero.

$$\frac{df'_x(s)}{dw_\tau^{[y]}} = \frac{d}{dw_\tau^{[y]}} \sum_{a=1}^{4} A'_a(s) w_a^{[x]} = 0 \tag{2.35}$$

### 2.9.2 Derivative of $E_{TLS}$

In this section, we take the derivative of $E_{TLS}$ with respect to the basis function weights. Later in the derivation we will have to specify which dimension ($x$ or $y$) the weight variable is associated with.

Recall the equation for computing $E_{TLS}$.

$$E_{TLS} = \int \vec{g}(s)^T H(\vec{f}(s))\vec{g}(s)ds \tag{2.36}$$

Notice, however, that the term $H(x, y)$ is only defined at pixel values, while $\vec{f}(s)$ is defined continuously. We use bilinear interpolation of $E_{TLS}$ for the points around the true location $\vec{f}(s)$. Define $\alpha$ and $\beta$ as the fractional part of $\vec{f}(s)$ which is needed for the bilinear interpolation and is illustrated in figure 2.14. Defining $x$ and $y$ as the integer parts of $\vec{f}(s)$, the equation for $E_{TLS}$ with bilinear interpolation can then be stated as follows.

$$\begin{aligned} E_{TLS} &= (1-\alpha)(1-\beta)\int \vec{g}(s)^T H(x,y)\vec{g}(s)ds \\ &+ \alpha(1-\beta)\int \vec{g}(s)^T H(x+1,y)\vec{g}(s)ds \\ &+ (1-\alpha)\beta\int \vec{g}(s)^T H(x,y+1)\vec{g}(s)ds \\ &+ \alpha\beta\int \vec{g}(s)^T H(x+1,y+1)\vec{g}(s)ds \end{aligned} \tag{2.37}$$

When the derivative of this equation is taken with respect to a basis function weight $w$ each term expands to two via the product rule.

$$\begin{aligned} \frac{d}{dw}E_{TLS} &= \frac{d}{dw}\{(1-\alpha)(1-\beta)\}\int \vec{g}(s)^T H(x,y)\vec{g}(s)ds \\ &+ (1-\alpha)(1-\beta)\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x,y)\vec{g}(s)ds\right\} \\ &+ \frac{d}{dw}\{\alpha(1-\beta)\}\int \vec{g}(s)^T H(x+1,y)\vec{g}(s)ds \\ &+ \alpha(1-\beta)\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x+1,y)\vec{g}(s)ds\right\} \\ &+ \frac{d}{dw}\{(1-\alpha)\beta\}\int \vec{g}(s)^T H(x,y+1)\vec{g}(s)ds \\ &+ (1-\alpha)\beta\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x,y+1)\vec{g}(s)ds\right\} \\ &+ \frac{d}{dw}\{\alpha\beta\}\int \vec{g}(s)^T H(x+1,y+1)\vec{g}(s)ds \\ &+ \alpha\beta\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x+1,y+1)\vec{g}(s)ds\right\} \end{aligned} \tag{2.38}$$

First we consider only the terms which take the derivative of $\alpha$ and $\beta$. Before giving the equation for this quantity, notice the following.

$$\alpha = f_x(s) - \lfloor f_x(s) \rfloor \tag{2.39}$$

$$\beta = f_y(s) - \lfloor f_y(s) \rfloor \tag{2.40}$$

We assume that both $x(s)$ and $y(s)$ are not integers and therefore the derivative of $\alpha$ and $\beta$ is defined. Taking the derivative of these with respect to an $x$-axis B-Spline weight, we obtain the following quantities.

$$\frac{d}{dw_\tau^{[x]}}\{\alpha\} = \frac{d}{dw_\tau^{[x]}}\left\{f_x(s)\right\} \tag{2.41}$$

As noted in equation 2.33 this is the single basis function corresponding to the weight $w_\tau^{[x]}$ and can be easily computed. Notice that $\beta$ is not dependent on x-axis related B-Spline weights.

$$\frac{d}{dw_\tau^{[x]}}\{\beta\} = 0 \tag{2.42}$$

We now consider the derivative of terms in equation 2.38 which have derivatives of $\alpha$ and $\beta$ in them. With respect to an x-axis B-Spline weight the terms are as follows.

$$\int \vec{g}(s)^T \begin{pmatrix} -(1-\beta)H(x\quad,y) \\ +(1-\beta)H(x+1,y) \\ -\beta H(x\quad,y+1) \\ +\beta H(x+1,y+1) \end{pmatrix} \vec{g}(s)\frac{df_x(s)}{dw_\tau^{[x]}}ds \tag{2.43}$$

Similarly, we can take the derivative with respect to a y-axis B-Spline weight.

$$\int \vec{g}(s)^T \begin{pmatrix} -(1-\alpha)H(x\quad,y) \\ -\alpha H(x+1,y) \\ +(1-\alpha)H(x\quad,y+1) \\ +\alpha H(x+1,y+1) \end{pmatrix} \vec{g}(s)\frac{df_y(s)}{dw_\tau^{[y]}}ds \tag{2.44}$$

We turn our attention now to the terms in equation 2.38 which take the derivative of the integral portion.

$$\begin{aligned}
&(1-\alpha)(1-\beta)\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x,y)\vec{g}(s)ds\right\} \\
&+\alpha(1-\beta)\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x+1,y)\vec{g}(s)ds\right\} \\
&+(1-\alpha)\beta\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x,y+1)\vec{g}(s)ds\right\} \\
&+\alpha\beta\frac{d}{dw}\left\{\int \vec{g}(s)^T H(x+1,y+1)\vec{g}(s)ds\right\}
\end{aligned} \tag{2.45}$$

Since the weight $w$ and the integration variable $s$ are independent, by the Leibniz integral rule we can bring the derivative within the integral. Furthermore, notice that the $H$ matrices are not directly dependent on the spline parameter $s$ as long as the spline continues interpolating from the same pixels. Finally, notice that the standard power rule and chain rule still work for quadratic symmetric matrix equations. We rewrite the equation as follows.

$$2\int \vec{g}(s)^T \begin{pmatrix} (1-\alpha)(1-\beta)H(x & y) \\ +\alpha(1-\beta)H(x+1 & y) \\ +(1-\alpha) & \beta H(x & y+1) \\ +\alpha & \beta H(x+1 & y+1) \end{pmatrix} \frac{d}{dw}\left\{\vec{g}(s)\right\} ds \tag{2.46}$$

The only remaining quantity we need to compute is the derivative of $\vec{g}(s)$. Recall the equation for $\vec{g}(s)$.

$$\vec{g}(s) = \begin{pmatrix} f'_x(s) \\ f'_y(s) \\ 1 \end{pmatrix} / \sqrt{f'_x(s)^2 + f'_y(s)^2 + 1^2} \tag{2.47}$$

The derivative of this quantity is computed via a product rule.

$$\begin{aligned}
\frac{d}{dw}\left\{\vec{g}(s)\right\} =& \frac{d}{dw}\left\{\begin{pmatrix} f'_x(s) \\ f'_y(s) \\ 1 \end{pmatrix}\right\}\left(f'_x(s)^2 + f'_y(s)^2 + 1^2\right)^{-\frac{1}{2}} \\
&+\begin{pmatrix} f'_x(s) \\ f'_y(s) \\ 1 \end{pmatrix}\frac{d}{dw}\left\{\left(f'_x(s)^2 + f'_y(s)^2 + 1^2\right)^{-\frac{1}{2}}\right\}
\end{aligned} \tag{2.48}$$

The first term is easily computed using equation 2.34. Notice, however, that the $y$ vector component will be zero when the derivative is with respect to an $x$-axis related B-Spline weight (and vice versa). The third component of the vector will always be zero. The second term further decomposes by the chain rule.

$$-\frac{1}{2}\begin{pmatrix} f'_x(s) \\ f'_y(s) \\ 1 \end{pmatrix}\left(f'_x(s)^2 + f'_y(s)^2 + 1^2\right)^{-\frac{3}{2}} \frac{d}{dw}\left\{\left(f'_x(s)^2 + f'_y(s)^2 + 1^2\right)\right\} \tag{2.49}$$

Finally, for an x-axis related weight term $w_\tau^{[x]}$ the last term becomes the following.

$$2f'_x(s)\frac{d}{dw_\tau^{[x]}}\{f'_x(s)\} \tag{2.50}$$

This is computed by equation 2.34. The case for a y-axis related weight term is similar.

Following these derivations we have been able to analytically differentiate $E_{TLS}$ with respect to the B-Spline weights taking into account both the bilinear interpolation present in the computation of $H$ and the homogeneous coordinates of $\vec{g}(s)$. This allows us to use fast optimization algorithms which require gradient information to solve for the optimal snake in application settings.

### 2.9.3 Derivative of $E_{velocity}$

In this section, we take the derivative of $E_{velocity}$ with respect to the basis function weights. Later in the derivation we will have to specify which dimension ($x$ or $y$) the weight variable is associated with.

$$\frac{dE_{velocity}}{dw} = \frac{d}{dw}\int \left\|\vec{f}'(s) - \vec{V}(\vec{f}(s))\right\| ds \tag{2.51}$$

Since the weight $w$ and the integration variable $s$ are independent, by the Leibniz integral rule we can bring the derivative within the integral.

$$\frac{dE_{velocity}}{dw} = \int \frac{d}{dw}\left\|\vec{f}'(s) - \vec{V}(\vec{f}(s))\right\| ds \tag{2.52}$$

Using the $\ell_2$ norm, this can be rewritten as

$$\frac{dE_{velocity}}{dw} = \int \frac{d}{dw}\sqrt{r_x + r_y}\; ds \tag{2.53}$$

where

$$\begin{aligned} r_x &= \left(f_x'(s) - V_x(\vec{f}(s))\right)^2 \\ r_y &= \left(f_y'(s) - V_y(\vec{f}(s))\right)^2 \end{aligned} \tag{2.54}$$

Focusing on the square root term, we compute the following using the chain rule.

$$\frac{d}{dw}\sqrt{r_x + r_y} = \frac{1}{2\sqrt{r_x + r_y}}\;\frac{d}{dw}(r_x + r_y) \tag{2.55}$$

Focusing on the term $r_x$ we compute

$$\frac{dr_x}{dw} = \frac{d}{dw}\left(f_x'(s) - V_x(\vec{f}(s))\right)^2 = 2\left(f_x'(s) - V_x(\vec{f}(s))\right)\left(\frac{df_x'(s)}{dw} - \frac{dV_x(\vec{f}(s))}{dw}\right) \tag{2.56}$$

The mixed second derivative of the B-Spline term was computed in equation 2.34 for the case of an $x$ related weight term $w^{[x]}$. If the weight is instead $y$ related ($w^{[y]}$) the resulting derivative is zero. We continue by taking the derivative of the vector field.

$$\frac{dV_x(\vec{f}(s))}{dw} = \frac{\partial V_x(\vec{f}(s))}{\partial f_x(s)}\frac{df_x(s)}{dw} + \frac{\partial V_x(\vec{f}(s))}{\partial f_y(s)}\frac{df_y(s)}{dw} \tag{2.57}$$

The $V$ terms are just the derivatives of the road vector field and can be computed by finite differences and interpolation at location $\vec{f}(s)$ as is done in figure 2.2. If the weight variable is $x$ related then the first term is non-zero while the second is zero. Combining these equations we are able to calculate the derivative of $r_x$. The derivative of $r_y$ can be derived in a similar fashion.

In this way, we are able to compute the derivative of the $E_{velocity}$ energy function with respect to the basis function weights. Although we have only shown how to differentiate two of the energy functions, every presented energy function is analytically differentiable and easily computable.

**Algorithm 1** Find all roads

**while** seed points available **do**

- Choose a seed point and initialize a short initial snake in the direction most consistent with the optic flow equation

**while** snake is allowed to grow (see section 2.10.2) **do**

- Find the snake parameters which minimize the energy function
- Add a span to one of the ends of the snake

**end while**

- Optimize the snake with the inclusion of the snake width energy functions $E_{thicker}$ and $E_{change-thick}$
- Remove the area around the snake and around the initial seed location from the set of allowable seed positions

**end while**

Figure 2.15: The set of seed locations for a traffic scene. The darker the pixel location, the more consistent the spatio-temporal derivatives in the region are and the better it is as a start location for a snake.

## 2.10 Snake Optimization Algorithm

We construct our snakes using a cubic B-Spline, which ensures continuity and differentiability of the snake and allows for easy differentiation of the energy functions. The snake is discretized using a uniform sampling with an average of one point per pixel that the snake covers. A control point is placed approximately every 10 pixels. Our method is outlined in algorithm 1. For our optimization algorithm we use the standard interior-point method provided by Matlab.

### 2.10.1 Snake Seed Locations

A seed set of possible initialization points is found by looking for locations of consistent motion. For each pixel with $C(x, y) \geq \gamma$ a small snake is placed with the velocity of best fit to $H(x, y)$ as determined by $E_{LS}$ and the energy value of the snake is recorded at that location. The first snake is initialized at the location with the lowest energy. After optimization of the snake all points covered by the snake and around the initialization point of the snake are removed from the seed set and the remaining point with the smallest energy value is chosen as the location for initializing the next snake. This procedure is repeated until there are no more points in the seed set. Figure 2.15 shows an image of the seed locations before the first snake is initialized.

### 2.10.2 Stopping Condition

When adding spans to the ends of the snake, we must evaluate whether the new section of the snake is still on the road. One stopping condition is when the snake reaches the edge of the image. The snake must also stop if it encounters spatio-temporal derivatives incompatible with its own speed and direction or if it encounters the end of the road. A simple threshold on the energy functions $E_{TLS}$ and $E_{off-road}$ for the newly added span, can effectively and robustly detect when the snake is on the road.

## 2.11 Results of Snake Models

In this section, we show the results of fitting snakes to video data. The NGSIM data set of Peachtree Street in Atlanta was captured through a collaboration of researchers interested in modeling vehicle behavior [2, 65]. A series of cameras were set up viewing consecutive parts of an urban road, and 15 minutes of data were simultaneously recorded in each. Extensive ground truth is freely available for the data set, including georeferenced lane centerlines, intersection locations and traffic light cycles.

Figure 2.17 shows snakes fit to the spatio-temporal derivatives computed from the NGSIM data. In all cases, the main roads in the scene were automatically discovered

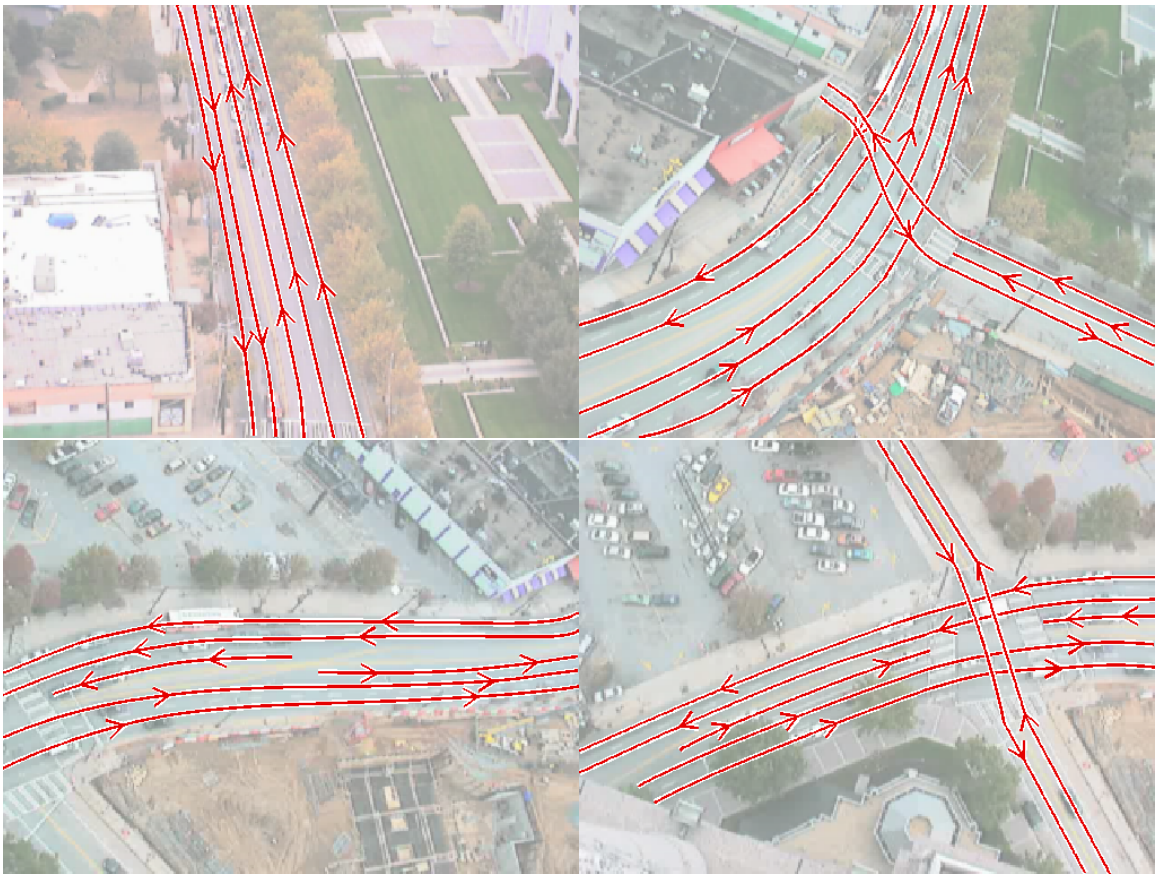
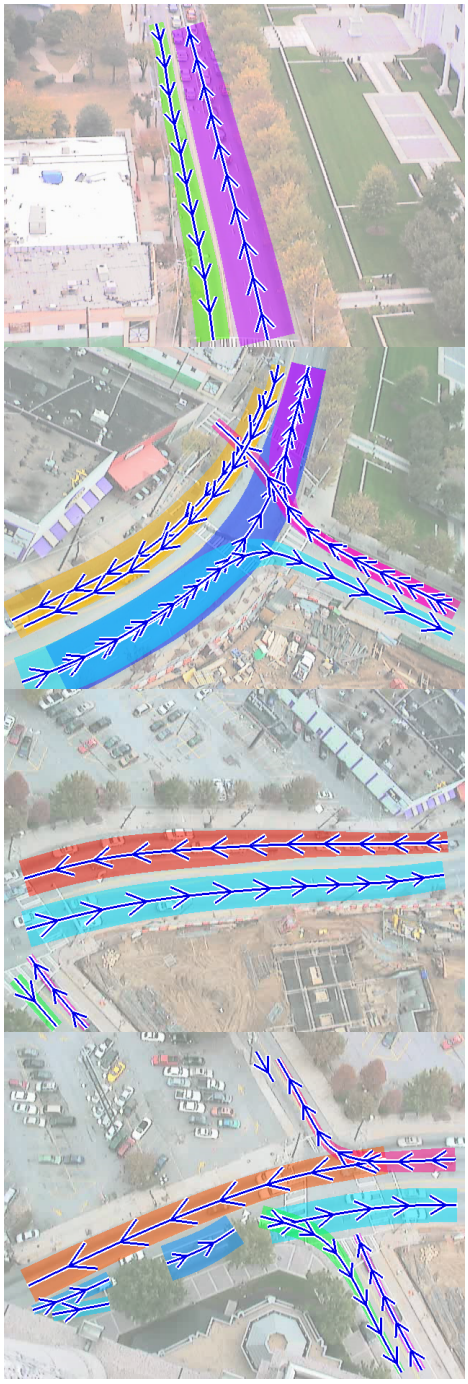


Figure 2.16: Hand annotated lanes from NGSIM, used as ground truth for evaluation.



| Mean Min Dist (pixels) | Mean Angle (degrees) |
|---|---|
| 1.4 | 1.2 |
| 3.6 | 1.5 |

| Mean Min Dist (pixels) | Mean Angle (degrees) |
|---|---|
| 5.6 | 9.7 |
| 0.9 | 3.6 |
| 2.5 | 7.4 |
| 2.5 | 7.8 |
| 2.9 | 16.0 |
| 1.9 | 5.3 |

| Mean Min Dist (pixels) | Mean Angle (degrees) |
|---|---|
| 2.9 | 1.8 |
| 2.1 | 1.8 |
| † 39.9 | † 73.9 |
| † 39.6 | † 107.8 |

| | |
|---|---|
| 5.0 | 16.0 |
| 3.7 | 7.8 |
| 3.3 | 15.9 |
| 1.9 | 10.3 |
| 4.1 | 4.7 |
| 1.1 | 2.6 |
| 1.8 | 6.3 |
| 0.4 | 1.7 |
| 3.1 | 15.2 |

Figure 2.17: Snakes, shown in blue with arrows indicating their speed are shown over a frame of the video. The area which each snake covers is shaded a distinct color based on its direction. The mean minimum distance to one of the hand annotated lanes of figure 2.16 is computed for each snake. The mean angular distance between the road direction and the direction of the closest hand annotated lane is also computed and shown for each road. Data marked with † is from snakes that fit to roads which were not annotated in the ground truth.

and parameterized with direction, speed, and width. There was no post-processing applied to the results to remove (or connect) roads, or to deal with the effects of occluding trees and buildings. For comparison, please refer to figure 2.16, which has the ground truth lane locations in the scene as determined by hand annotation. We report the mean distance of the road centerline to the nearest ground truth lane centerline and the mean angle between the tangent of the snake and the direction of the nearest lane centerline.

As with most snake implementations, the relative weighting of the energy functions must be empirically determined. The final energy function which is used on all the videos in figure 2.17 is as follows.

$$\begin{aligned} Etotal = & 60E_{TLS} + 0.1E_{smooth} + 1E_{off-road} \\ & + 1E_{minimum-speed} + 0.05E_{thicker} + 0.1E_{change-thick} \end{aligned} \tag{2.58}$$

To show the robustness of the parameter choices, figure 2.18 shows results for the same scene when each energy function weight is multiplied by a factor of $\frac{1}{2}$ and 2. There is very little difference between these results, showing that the exact value of the parameters is not particularly important: the algorithm will determine the correct lane annotations over a wide range of parameter values.

Although $E_{TLS}$ is the most effective of the velocity matching energy functions, since it is nicely bounded within a small range, we include results when using the other velocity matching energy functions in figure 2.19. All energy functions are able to parameterize all major roads; however, there are some differences. $E_{TLS}$ is able to make use of the ambiguity in the structure tensor to have multiple roads take different paths through the intersection. $E_{LS}$ does not find the correct width for the two roads on the right side of the scene, this is due to a scaling issue between the scaling factors for $E_{LS}$ and $E_{thicker}$. Such problems are hard to avoid when using $E_{LS}$ since it has an unpredictable range of values. $E_{normal-flow}$ is also nicely bounded, and takes two different paths through the intersection. One of the roads has an end which becomes very narrow, and an extra, small road is created in the upper portion of the scene. In general $E_{normal-flow}$ is not as effective as $E_{TLS}$. $E_{velocity}$ does not have access to the ambiguity captured within the spatio-temporal structure tensor and therefore behaves in a much more rigid fashion. It is only able to take one path through an intersection, and in general has problems with finding the correct width of the road.

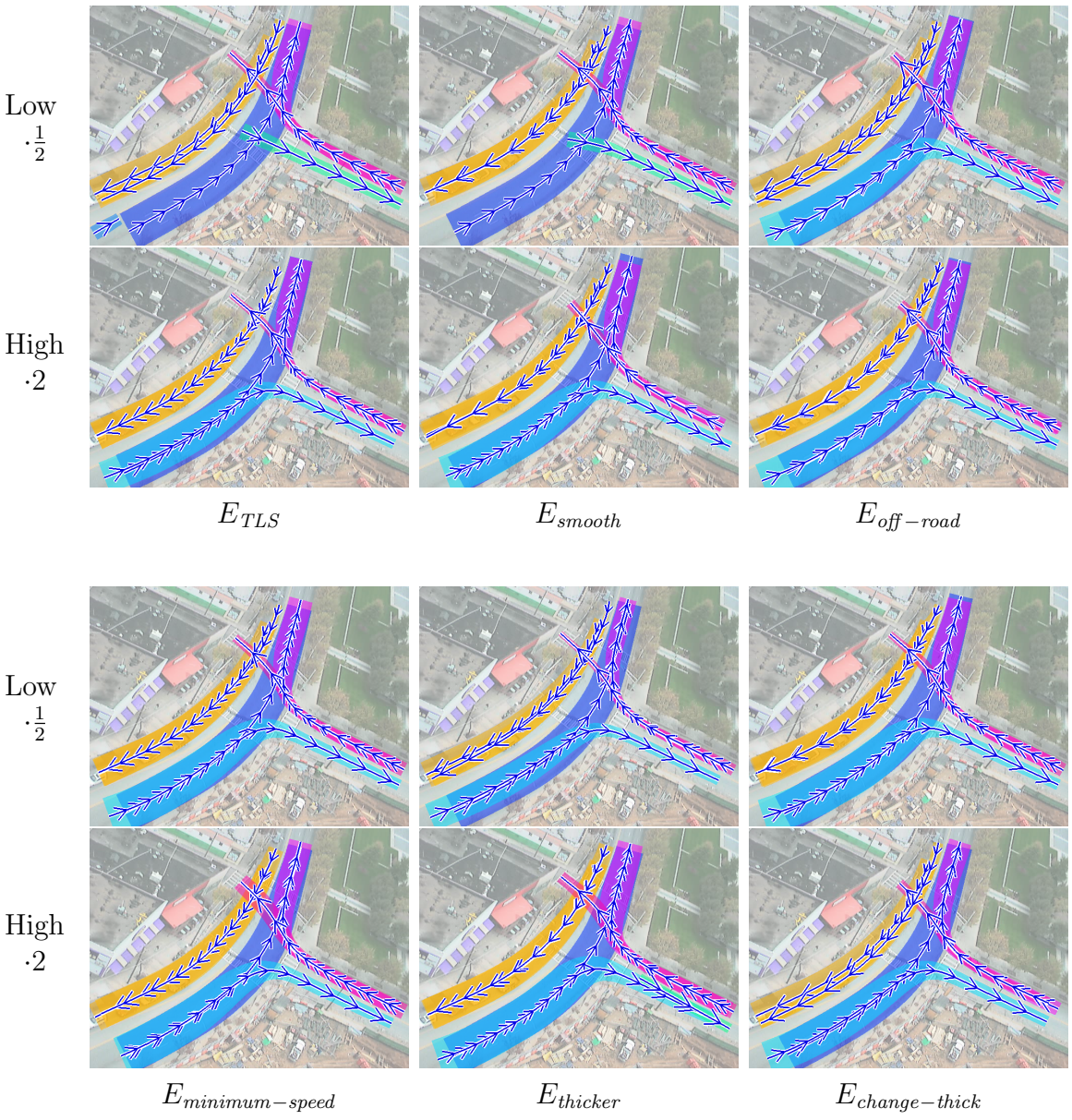
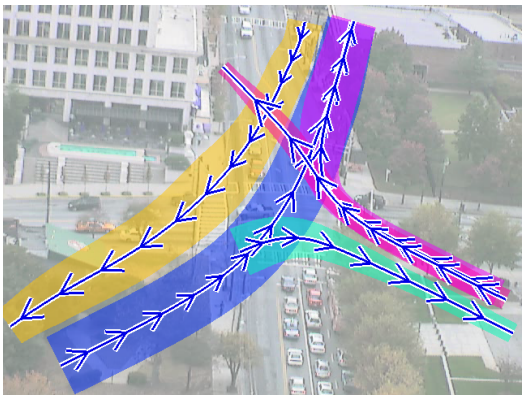
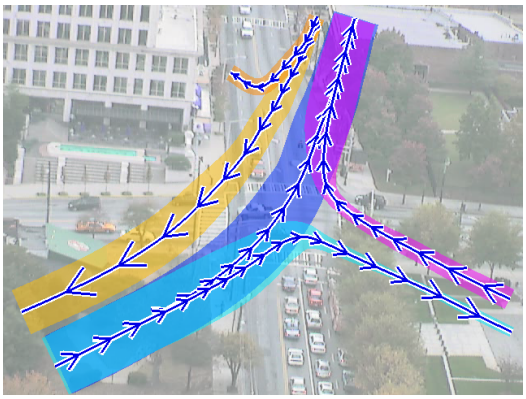


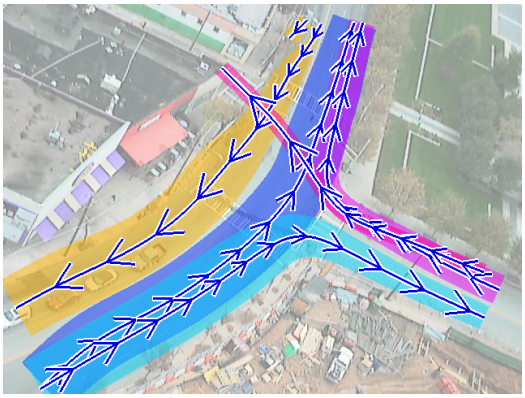

Figure 2.18: For the weight assigned to each energy function we divide it by a factor of 2 from the default value on top and multiply it by a factor of 2 on bottom. Although there are subtle differences in the results, the main roads are still extracted in all cases.



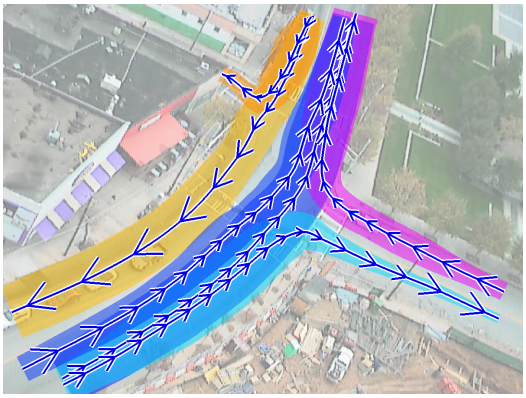
$E_{TLS}$



$E_{LS}$



$E_{normal-flow}$



$E_{velocity}$

Figure 2.19: Various velocity matching energy functions are used to compute a road parameterization. In general, $E_{TLS}$ is the easiest energy function to control due to its bounded range. $E_{LS}$ shrinks to a very small road width on the right side. This is caused by a scaling issue between the scaling factors for $E_{LS}$ and $E_{thicker}$, which is difficult to avoid due to the unpredictable range of the $E_{LS}$ energy function. The three energy functions based on matching the snake derivative to the spatio-temporal derivative statistics directly are more effective within intersections, where the inherent ambiguity of the data allows different roads to be placed in different directions. The energy function $E_{velocity}$ is very rigid in how it parameterizes the data, always going the same direction in intersections and having trouble finding the correct road width.

Although this energy function is not strictly bounded between any values, it tends to remain within a small range and is not too difficult to scale properly.

## 2.12 Conclusion

The spatio-temporal structure tensor is a convenient, efficient, and informative summary of the image gradient information captured over time at a particular pixel. While it has been used in several applications before, there has been no clear illustration of failure modes, recognition that different methods of computing the flow vector give noticeably different results, or integration of the structure tensor into a larger scale fitting problem.

Our results with fitting vectors to spatio-temporal derivatives show that using least-squares is best for solving for optic flow in video sequences. The structure tensor field itself is a valuable characterization of motion patterns which we use directly as a basis for fitting parametric snake models to roads.

We define new snake energy functions based on different methods of matching a velocity vector to spatio-temporal derivative statistics. This allows us to construct stable open-ended snakes with constrained derivative terms. Additionally, we add the ability to adjust the thickness of the snake to find the proper width of the road.

Areas for future work include integrating the parametric roads as priors within a tracking algorithm and solving for an explicit road network. Additionally, one can use the parametric models found in locations with significant road motion to learn a scene-specific appearance model in order to extend roads onto areas that have not yet observed motion.

# Chapter 3

# Parameterizing and Modeling Tissue Motion within Medical Images


## Abstract

*Medical imaging techniques such as CT and MRI are becoming increasingly common for diagnosis and treatment. Tissue motion during image acquisition can cause artifacts and incorrect quantitative measurements if not properly taken into account. In this chapter we look at three techniques for modeling tissue motion within medical images. First, we present a traditional approach to 4D CT lung modeling based on an external measurement of breath phase. Second, we present our alternative data driven approach which does not require an external breath measurement. Third, we present a novel free form deformation based tissue motion model. This approach is applicable to a more general set of tissue motion modeling problems and will be presented using a heart/lung MRI image dataset and synthetic data.*


## 3.1 Introduction

Medical imaging gives a 3D view of tissues in the body. Often, the goal is to understand the motion of the tissues. For CT and MRI data, the key problem is that these imaging tools cannot directly measure motion, do not capture a complete 3D image

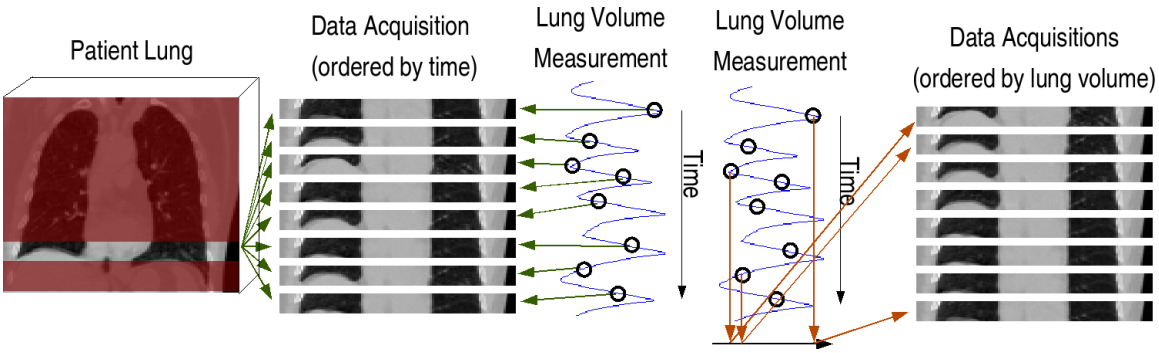



Figure 3.1: A CT scanner is used to repeatedly image slabs of the lung approximately 1 inch in height while a patient breaths. At the same time a belt around the chest of the patient measures lung volume. Although, the data acquisitions, which are ordered by time do not have a coherent pattern with respect to breathing, the slabs can be reordered by the belt measurement.

at once, and do not capture images frequently enough to directly track tissue motion. In this chapter we consider two approaches to address this problem.

CT imaging is often used in lung cancer diagnosis and treatment. Unfortunately, lung tumors move with the lung. Since radiation treatments affect both the tumor and the healthy tissues near the tumor, effective radiation treatment planning requires modeling the motion of the tumor and the surrounding healthy tissues. In the past, motion artifacts have usually either been ignored or reduced through the use of breath holds during image acquisition. However, for many patients, it is infeasible to maintain a breath hold during radiation treatment. Accordingly, radiation treatment plans create large margins around the tumor to account for breathing, and this causes extra damage to healthy tissue. Thus, to make good radiation treatment plans, it is necessary to understand the motion of the tumor and surrounding tissue during breathing.

Simple solutions to the motion issue include using fluoroscopy to monitor tumor movement by placing a radioactively visible marker near the tumor and using fixed, breath hold images for treatment planning. In some cases, patient breathing has been controlled in an attempt to predict tumor position and create motion maps [75, 99]. Recently, four-dimensional CT (4D CT) has been developed, which uses an external measure of breath phase to create multiple CT datasets at various breath phases [28, 38, 40, 89]. This is more effective than fluoroscopy at determining tumor

movement [87]. This technology continues to advance in complexity, but the need for an external measure of breathing has remained a key component of all lung 4D CT methods thus far.

The traditional way of creating a 4D CT lung model uses the following general procedure [40, 41]. Figure 3.2 shows the parts of the traditional model along its right side.

1. *Image Acquisition* - A common multi-slice clinical CT scanner is configured to capture a volume roughly one inch in height. As shown in figure 3.1, a cross sectional volume of the lung usually about one inch in height is imaged repeatedly while the patient breaths. The patient is then moved one inch through the CT scanner, to the next couch position, and more images are acquired. We use the term *slab* to refer to a single image acquisition, taken at a particular couch position and breath phase.

2. *Breath Phase Estimation* - Each slab takes approximately one second to acquire with several seconds between acquisitions. This is slow enough and patient breathing is irregular enough that breathing cannot be estimated from the time of image acquisition. Traditionally, an external breath measurement such as a belt measuring chest circumference is used to estimate the air volume in the patient's lung. Although breathing is a cyclic process, since lung images with the same air volume look very similar during inhalation and exhalation, breath phase is generally modeled as a one dimensional parameter which can be thought of as air volume in the lung.

3. *Lung Volume Reconstruction at All Breath Phases* - A complete 3D data volume is generated for each part of the breathing cycle by picking a target breath phase and choosing the data at each couch position acquired closest to the target breath phase. Figure 3.3 shows, on the left side, the breath phase measurements for each slab plotted against the CT scanner position. For a target breath phase shown as a vertical line, the right side shows the reconstruction of the lung corresponding to the slabs closest to the target breath phase.

This procedure, although used clinically, has substantial limitations. Errors such as the diaphragm appearing twice in the lower part of figure 3.3, are common. These problems arise from the fact that the measurement of the chest circumference is not

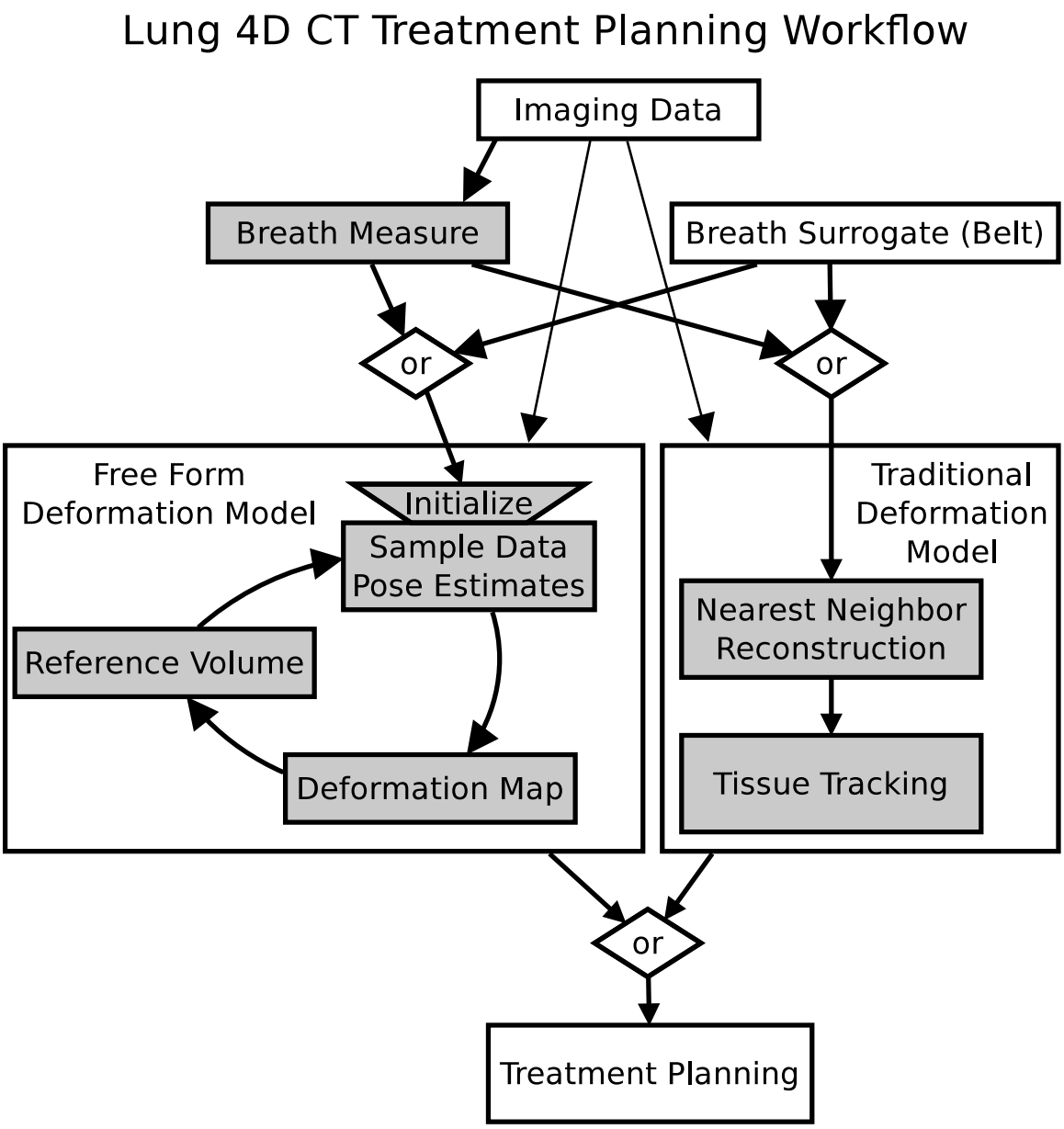



Figure 3.2: Two alternative workflows for using 4D CT lung reconstructions. Boxes marked in gray are methods which are presented in the text. Methods for estimating the breath measure are shown in section 3.4, the free form deformation model is in section 3.5. The rightmost path in this diagram constitutes the standard approach and is described in section 3.3.

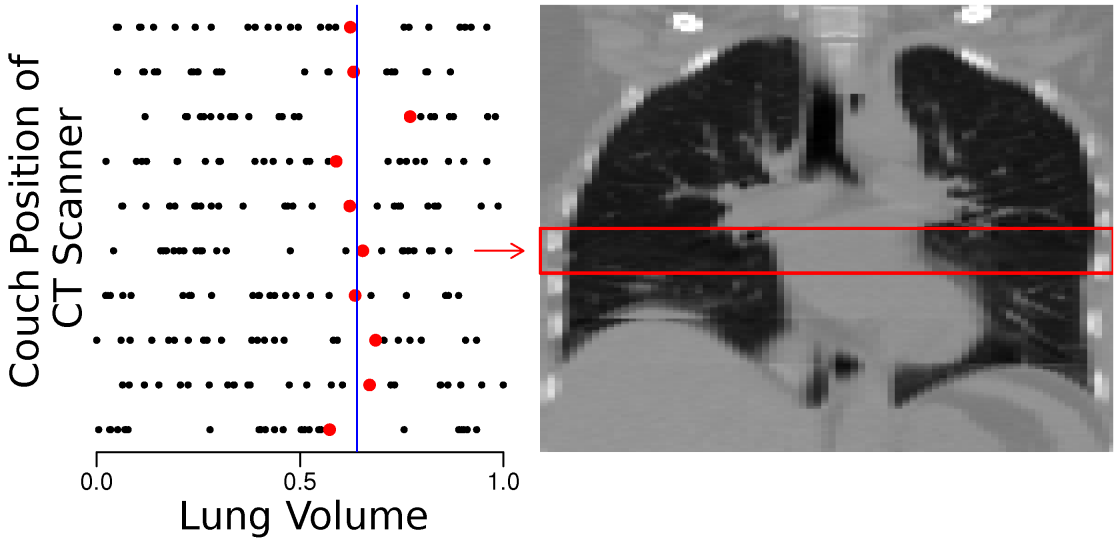



Figure 3.3: On the left, the external breath measurements for each 1 inch high image acquisition is shown organized by scanner couch position. For a target breath phase, shown as a vertical line, a lung volume is created by picking the slab at each couch position of the CT scanner with breath phase closest to the line. On the right, a coronal cross section of the 3D lung reconstruction corresponding to the target breath phase is shown. The red box shows the size of image acquisitions at a single couch position (a slab). Several artifacts caused by tissue motion are visible near the diaphragm.

perfectly correlated with lung air volume (or the phase of the breathing cycle). In addition, not all breath phases have a corresponding sample image. Section 3.3 details the complete standard process, including several problems that commonly arise.

In section 3.4 we present a novel method for assembling 4D CT datasets by creating a substitute for the external breath measurement which is based on the image data itself. This simplifies image acquisition, data synchronization, and allows 4D CT to be used retrospectively even when an external breath measurement was not initially acquired. It is our intention that any algorithm that would use an external physiological measurement can instead be modified to use our breath measure. This portion of the work was first published in [35, 37].

The goal of lung 4D CT is generally to understand tumor motion in the lung during breathing. In addition to the lung reconstructions at an arbitrary breath phase, this requires knowledge of how the tissue itself is moving during breathing. In the traditional approach, a small number of lung reconstructions are created at evenly spaced intervals in the breath phase. Techniques such as optic flow [38] and template matching [35] can be used to compute tissue motion by following motion through these lung reconstructions.

In section 3.5 we construct a richer 4D CT lung model by directly modeling tissue deformation and generalizing to different imaging modalities such as MRI and to situations where more than one degree of freedom is present. A three part model is used where the following problems are simultaneously solved.

1. *Reference volume estimation* – Estimating the undeformed appearance of the data volume.

2. *Parameterized deformation map generation* – Estimating the deformation of the reference volume due to tissue motion.

3. *Deformable-pose estimation* – Estimating, for each time sample, the motion parameter of the deformation map at the time of sampling.

Our approach is data driven and does not rely on either a separately acquired reference volume or feature detection and tracking. Tissue motion is modeled by deforming a

reference volume with a cubic B-Spline free form deformation. An iterative method is used to simultaneously solve for the reference volume and deformation map while updating the pose estimates. This same process is demonstrated on 4D CT lung data and heart/lung cine-MRI data.

The remainder of this chapter is divided into three parts. First, we present the traditional method for creating 4D CT lung models using externally measured breath phases. Second, we present a method of creating a global breath measure without externally measuring breath phases. Third, we present an improved method for creating a lung deformation model based on a free form deformation map and a reference lung. We show this approach also applies to cine-MRI of a heart and evaluate on simulated data. Figure 3.2 shows a flowchart of the different parts and approaches to creating a 4D CT lung model.

## 3.2 Related Work

The use of 4D aware imaging and radiation treatment is a growing area with many applications (see [53] for a review). Due to a diversity of scanner technologies, 4D CT of the lung can be executed in a number of different ways. Some examples include spiral and cine mode CT with an external breath measurement [40, 89]. On a lower level, the X-ray beam can be gated based on a physiological signal and specialized CT algorithms used to reconstruct the tissue volume [66].

The addition of a deformation model to 4D CT and cardiac MRI has been explored previously. B-Spline models of tissue deformation have been fit to tagged MR images [44, 85], and untagged MR images [18]. In 4D CT of the lung a B-Spline deformable model has been used to find the deformation between 3D reconstructions at approximate maximum inhale and approximately maximum exhale [78]. More recent work aims to minimize the artifacts caused in the 3D reconstruction at each breath phase. When data is missing in a particular slab, optic flow is used to reconstruct a 4D CT data set through interpolation of existing measurements to the required respiratory phase. This helps for some kinds of artifacts but not artifacts caused by vertical motion that causes tissue to leave the (one inch high) data acquisition volume [28].

The newest methods for creating 4D CT reconstructions of lung motion directly infer a deformation map from the slab data. Some 4D CT studies of the lung have relied on a given reference image which is acquired by scanning the patient during a breath hold [58, 102]. A lung deformation model can be created by registering the sample images from different parts of the breath to the breath hold image and fitting a B-Spline to the deformation. McClelland *et al.* also briefly investigated the creation of a breath measure from the slab data by measuring the anterior-posterior movement of the skin surface in the slab [58]. A sine wave is fit to this signal to create a breath measure. Ultimately this method was abandoned in favor of an external breath marker. Our method circumvents both these problems, it acquires a breath measure directly from the CT images and does not require a separately acquired reference image.

Our deformation based motion model, which simultaneously aligns all images in a dataset, is similar to, but more general than, image registration, in which the transformation between two images is found. In particular, variational methods in computational anatomy find the geodesic path between images within the set of diffeomorphisms [8, 27, 60]. These transformations create a time parameterized vector field similar to our motion model. In this model, the deformation map and "time" magnitude is proportional to the amount of deformation needed to align the two images, which elegantly defines a metric between image pairs based on aligning the images with a diffeomorphism. On the other hand, in our model the deformation map and "time" magnitude (or breath phase magnitude) creates a physiologically meaningful vector field which describe tissue motion at every part of the breath cycle.

In the computer vision community, recent work on holistic analysis of sets of images that vary due to a few parameters is often based on manifold learning. Approaches include using manifold constraints to regularize segmentation of cardiac MR images [104], and to learn the low-dimensional parameterization of a cardiac MR set [82], in order to find nearby neighbors for effective image interpolation. However, neither approach solves for an explicit reference volume or deformation map.

The work in this chapter is related to manifold pursuit [80], which augments PCA by introducing invariance to an image transformation such as translation. This allows a set of images to be represented using aligned basis images and a transformation

function. Our work makes the image deformation much more general, using B-Spline deformation as opposed to affine warps, but assumes a single reference image rather than a linear subspace.

## 3.3 Traditional Approach to Lung 4D CT

To place our contributions in context, we explain the existing protocol in use at Washington University in St. Louis Radiation Oncology. The traditional approach to 4D CT uses an external belt measurement to label each image acquisition with a breath phase. A lung reconstruction at a particular breath phase is then created by combining slabs with similar breath phases and changes are extracted by tracking tissue motion through reconstructions.

### 3.3.1 4D CT Lung Data

4D CT image acquisition protocols can vary considerably; however, a necessary component is that there be multiple images of the same lung area during patient breathing. In our image acquisition protocol we acquired images from 25 lung cancer patients. Transverse volumes of the lung were acquired using a 16 slice CT scanner in cine mode while the patient was free breathing. At each couch position, 25 slabs (512 by 512 by 16) were collected consecutively, before moving the patient to the next couch position and collecting 25 slabs of the next section of lung. Since slabs were acquired during multiple breaths with each scan separated by several seconds, they are effectively unordered with respect to breath phase. Manual inspection was used to crop slabs to the smallest rectangular volume which contained lung tissue at any point in the breath cycle. The transverse plane was down-sampled from the original by a factor of 3. There was no downsampling in the superior-inferior axis. On average the downsampled data volume used for processing was 70 by 130 by 176 voxels.

### 3.3.2 Breath Phase of Slabs

During image acquisition, an external breath measure was collected using a belt measuring chest circumference. Figure 3.4(a) shows the belt measurement for each image: the x-axis shows the position of the slab in the lung, while the y-axis gives the value of the belt measurement for a particular slab. The size and hue of the dot corresponds to the value of the belt measurement and therefore, in this case, is exactly equivalent to the y-axis.

### 3.3.3 Constructing Lung Volumes

To construct a 4D lung model each slab must be labeled with its breath phase. A lung image of any part of the breath cycle can be generated by selecting a slab from each couch position with similar breath phase (as previously shown in figure 3.3).

We generate a discrete model of the 4D lung by creating 8 volumes at equally spaced intervals through the breath space starting from exhalation and going to inhalation. The distribution of how long a patient spends in each part of the breath is calculated by counting the number of slabs from each couch position whose breath measure value is closest to each volume. We label the volume with the largest number of slabs associated with it as the central volume. This volume tends to be near exhalation, since people tend to spend a longer portion of their breath cycle near an exhaled state.

We create a mask for each volume which specifies whether a specific voxel is part of the lung or not. A simple threshold of all voxels below the value of -200 Hounsfield Units (HU) partitions air filled spaces from denser tissue. To distinguish external air from internal air we estimate a location within each lung and flood fill all voxels connected to these points. This gives an initial mask of the lung which excludes outside air and small air filled structures within the lung. Image morphology operations are used to include the small air filled structures. The mask is dilated[1] and then immediately

[1]dilated – a new volume is created where every voxel is the maximum of its neighboring voxels in the old volume. Hence a voxel is in the mask if all of its neighbors in the old volume were in the mask.

eroded[2] by a small circular operator. This procedure has the effect of smoothing edges and removing small misclassified sections within the lung. All subsequent operations use this mask so as not to average voxels inside the lung with those outside of it.

### 3.3.4 Tracking Tissue Motion

A model of lung tissue movement during breathing can be generated by tracking motion through the reconstructed volumes. A template tracking approach is used between adjacent pairs of reconstructed volumes. We place a grid over the source volume $I_s$ and track tissue motion at each node using a 7 by 7 by 7 template centered at the node location $\vec{\ell} = (\ell_1, \ell_2, \ell_3)$. The template is compared to volume patches around the same location in the target volume $I_t$. We find the vector $\vec{v}$, within a small search space of $\pm 3$ by $\pm 3$ by $\pm 4$, which minimizes the sum of squared difference between the template and the volume patch at location $\vec{\ell} - \vec{v}$.

$$\arg\min_{\vec{v}} \sum_{k_1=-3}^{3} \sum_{k_2=-3}^{3} \sum_{k_3=-3}^{3} \left( I_s(\vec{\ell} + \vec{k}) - I_t(\vec{\ell} + \vec{k} - \vec{v}) \right)^2 \tag{3.1}$$

The vector $\vec{v}$ is the movement of the tissue centered at $\vec{\ell}$ in the source volume. During this tracking we only consider voxels that are within the mask of the lung and normalize the sum by the number of voxels which are part of the lung.

To track tissue throughout all the reconstructed volumes we track the tissue movement between pairs of reconstructed volumes moving outward from the central volume. First, the central volume is set as the source volume and an adjacent neighbor is the target volume. After tissue is tracked in this pair, the old target volume is set as the source volume and the next adjacent volume further from the central volume is set as the target volume. This continues, moving outward from the central volume, until tissue movement has been tracked throughout all reconstructed volumes. Figure 3.4(b) shows the movement vectors in a coronal cross-section of the lung near an inhaled state.

[2]eroded – a new volume is created where every voxel is the minimum of its neighboring voxels in the old volume.

The tracking algorithm has difficulties in untextured parts of the lung and in regions displaying the aperture problem, where an edge in the small template region can be matched to anywhere along a large edge. Additionally, errors in the lung volume reconstruction stage cause template tracking errors. These problems can be addressed using the more comprehensive deformation model which we introduce in section 3.5.

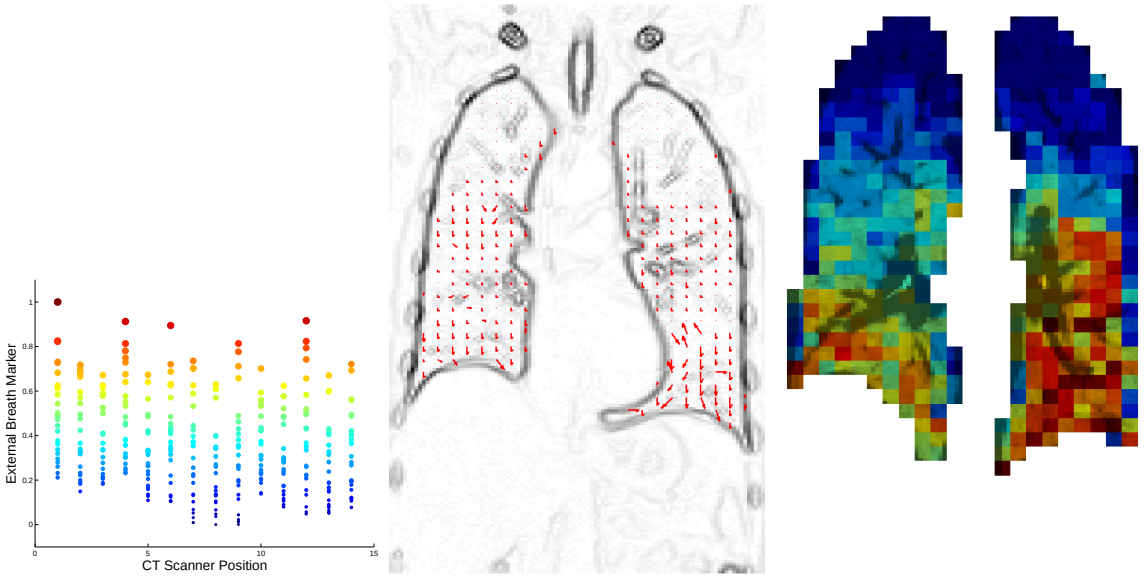
### 3.3.5 Magnitude of Motion

Tissue movement during the entire breath is calculated by placing the vectors from each reconstructed volume end to end and following them from the central volume outward. The end point of one motion vector is taken as the starting point in the next volume. Linear interpolation is used to determine the motion vector at points that are not directly on the grid. An aggregate vector is created by adding all the vectors along the path from the central volume to the inhaled state and subtracting the path from the central volume to the exhaled state. This vector gives the total movement from exhalation to inhalation of the tissue at a specific location in the central volume. The magnitude of this vector measures how much movement is present in each part of the lung. Figure 3.4(c) shows a heat map of this movement magnitude overlaid on the lung.

## 3.4 Breath Measure from Imaging Data

The 4D CT lung model presented in the previous section requires an external breath phase measurement for each slab. In this section, we present an alternative approach in which a breath measure is automatically computed directly from the CT imaging data. The estimation of the breath phase of a slab requires several parts.

1. *Local Breath Measure* - Each couch position is analyzed independently to determine relative positions in the breath cycle for each slab. We call this a local breath measure. The local breath measures in each couch position are in a separate coordinate system, which can have an arbitrary orientation and scale.




(a) External Breath Marker (b) Motion Vectors (c) Total Motion Magnitude

Figure 3.4: The various parts of a traditional 4D CT Lung model (patient 38). (a) The breath phase (lung volume) as determined by a belt around the patient's chest. (b) Vectors showing tissue motion at a breath phase near inhalation. (c) Coronal cross-section of lung color coded by the total motion magnitude at a particular location: red sections move more than blue sections.

The remaining steps are necessary to create a global breath measure, which is consistent over the entire lung, from the local breath measures.

2. *Identify Slabs From the Same Breath Phase* - To align local breath measures we must identify which slabs in neighboring couch position have the same breath phases. This is accomplished by measuring the boundary discontinuity between slabs in neighboring couch positions and assuming that slabs with little discontinuity are from the same breath phase.

3. *Breath Measure Orientation* - We orient the local breath measures in a consistent manner such that high values correspond to an inhaled state.

4. *Determining the Optimal Neighbor Alignment* - An alignment between the local breath measure of one couch position to its neighboring couch position is created by finding the linear transformation which best places each slab close to neighboring slabs with the same breath phase.

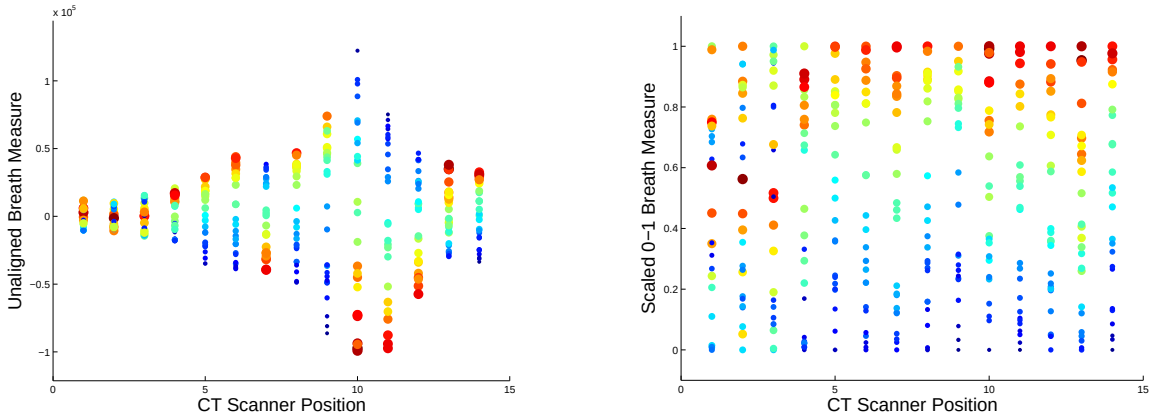
5. *Global Alignment* - A global alignment of all couch positions is created by transforming the local breath measure of each couch position into a single global coordinate system.

### 3.4.1 Measuring Breath Phase Within a Couch Position

Although breathing is a cyclic process, we find that images of the lung at the same lung volume are very similar whether the patient is inhaling or exhaling. This allows us to characterize the breath with a single parameter: lung air volume. We apply the Isomap algorithm as described in [84] to the slabs captured at a single couch position, using the square root of the sum of squared distance (using the k-nearest neighbors as a neighborhood criteria, with a value of 8 for k). We use the 1D parameterization produced by Isomap as the local breath measure for slabs computed at a given couch position. This breath measure is termed $f_i$, and defined as the mapping (computed through Isomap):

$$f_i : S_i \rightarrow \mathbb{R} \tag{3.2}$$

where $S_i$ is the set of slabs at couch position $i$.




(a) (b)

Figure 3.5: (a) 1 dimensional Isomap parameterization for each couch position of a single patient. The size and color of the circles represents their externally measured belt value. A global alignment of these local breath measures is crucial since each couch position may have a different scale and orientation. (b) 1 dimensional Isomap parameterization correctly oriented and scaled to be in the range [0,1]. The Isomap parameterization and belt measurements are more similar in the lower portion of the lung (scanner position 11) where there is more motion due to breathing.

Figure 3.5(a) shows the local breath measure coordinates generated by Isomap, shown for different couch positions (labeled on the x-axis). Ground truth estimates (measured by the extension of a belt around the chest) are depicted by the size and hue of the points in the figure. This figure highlights that the Isomap parameter has a high correlation with the ground truth, but the Isomap parameters are sometimes oriented differently, and the scale also varies. This is because the Isomap algorithm parameterizes the slabs based on their similarity to each other: there is no global reference to orient the parameterization, and different parts of the lung have different overall contrasts changes, creating different relative scales.

### 3.4.2 Global Breath Measure

Since the local breath measure is only meaningful within a single couch position, extra work is needed to align the local breath measures to create a globally consistent breath measure valid over the entire lung. We align local breath measures by a linear transform to create a single global breath measure.

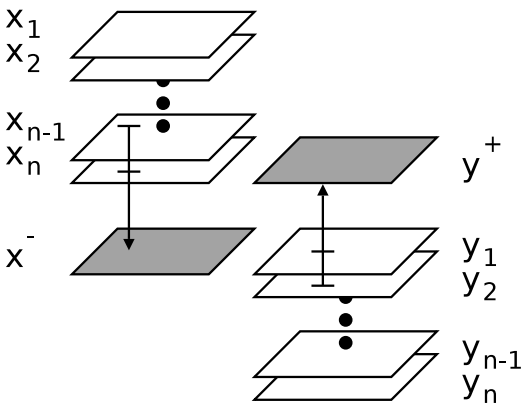
As an example of one way of scaling the local breath measures to create a global breath measure, we naively rescale the range of each local breath measure to be 0 to 1 as shown in figure 3.5(b). If each couch position has a slab from both complete inhalation and complete exhalation, this would create a consistent global breath measure. Unfortunately, patients do not breath in a perfectly regular pattern, and for some couch positions the patients do not reach maximum inhale or exhale. We create a more effective algorithm by comparing the junction between slabs in neighboring couch positions. When the slabs were acquired during similar breath phases, their junction should be smooth. We quantify this smoothness with a discontinuity measure.

We create a global breath measure in three parts. First, we use the discontinuity measure to find for each slab the two slabs in neighboring couch positions which are most likely to be from the same breath phase. Second, the local breath measure in each couch position is oriented to be from exhalation to inhalation. Third, we find the affine transformation which optimally aligns the local breath measures so as to minimize the distance from one slab to its two neighbors in the neighboring couch position.

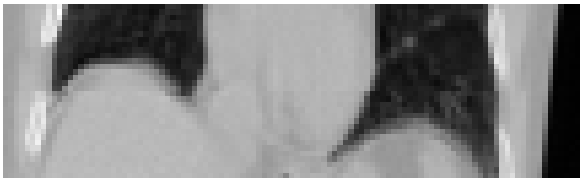
#### 3.4.2.1 Measuring Boundary Discontinuities

Slabs with similar breath phases from neighboring couch positions should have little discontinuity at their junction. Figure 3.6 shows two pairs of slabs. One pair matches well and is likely to come from the same breath phase while the other pair has a very discontinuous junction and is not likely to come from the same breath phase.

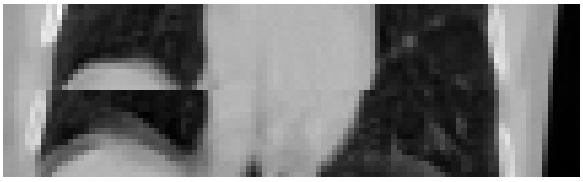
We denote the set of slabs within the $i$th couch position from the top as $S_i$. Given two slabs $x \in S_1$ and $y \in S_2$ we compute a discontinuity measure over the boundary between the slabs. We number the transverse slices which comprise the slab $x$ from top to bottom as $x_1, x_2, \cdots, x_n$. The discontinuity measure is computed by predicting the slice directly below the upper slab and comparing it to the top slice of the bottom slab and vice versa as shown in figure 3.6(a). The bottom two slices of $x$ are used to linearly predict the slice below them which we denote as $x^-$ ($x^- = 2x_n - x_{n-1}$). This slice is then compared to the top most slice in $y$. Similarly, the top two slices in $y$




(a) Discontinuity Measure ($\rho$)



(b) Example Low Discontinuity: $\rho = 18$



(c) Example High Discontinuity: $\rho = 42$

Figure 3.6: (a) The discontinuity measure ($\rho$) between two slabs is created by using the lower slices of the upper slab to predict what the slice directly below it will look like ($x^-$) and comparing that slice to the top slice of the bottom slab ($y_1$). The reverse computation of comparing $y^+$ and $x_n$ is also performed and the average taken. Coronal cross sections of two slabs from adjacent couch positions that come from (b) the same breath phase (c) different breath phases as determined by the discontinuity measure.

are used to predict the slice above them ($y^+ = 2y_1 - y_2$) and this is compared to the bottom most slice of $x$. The discontinuity measure $\rho$ is the sum of the $\ell_2$ distance between the predictions and the actual slices from the adjacent slab (this is the sum of squared error of voxel differences).

$$\rho(x, y) = \|x^- - y_1\| + \|y^+ - x_n\| \tag{3.3}$$

A small discontinuity measure $\rho$ as seen in figure 3.6(b) gives evidence that the two slabs were acquired at similar breath phases.

We use the discontinuity measure to create a paired list of matching slabs which are likely to come from the same breath phase. For each slab $x \in S_1$ the two slabs in $S_2$, $y_{x,1}$ and $y_{x,2}$, which minimize the discontinuity measure are found and the pairs $(x, y_{x,1})$ and $(x, y_{x,2})$ are added to the matching slabs list. Conversely, for each slab $y \in S_2$ the two slabs in $S_1$ which are least discontinuous with $y$ are found and the pairs $(x_{y,1}, y)$ and $(x_{y,2}, y)$ also added to the matching slabs list. This paired list of matching slabs is used to orient and align the local breath measures. Although we

presented the algorithm using two matching slabs, we evaluate using between 1 and 5 matching slabs in section 3.4.3.3.

#### 3.4.2.2 Orienting the Local Breath Measures

The local breath measure of each couch position has an uncertain orientation with respect to inhalation and exhalation. To avoid undesirable local minima in the optimization process, we explicitly orient the local breath measures before optimization. We first determine the orientation of the local breath measure at the diaphragm, where it is easy to compute due to the obvious changes caused by the diaphragm's motion. This orientation is then propagated outward to orient all local breath measures.

Since the largest contrast change in the lung images is that between the air filled lung and the surrounding tissue, the couch position with the greatest contrast change will be the one which sees the most movement of the diaphragm. At this couch position, the inhaled position, which contains the least diaphragm tissue, has a low average voxel value. In the exhaled position, more diaphragm tissue is visible and the average voxel value will be high. We orient the local breath measure such that a small breath measure value corresponds to an exhaled state.

The following method is used to explicitly orient all local breath measures to be consistent with that of the couch position with the diaphragm. For two adjacent couch positions, determine the correlation between the local breath measures of all pairs of slabs in the matching slabs list computed in the previous section. If this correlation is negative, then the local breath measure must be inverted to preserve a consistent global orientation in which the matching slabs have positively correlated local breath measure values. Moving outward from the couch position with the most diaphragm movement, this procedure is repeatedly applied until all local breath measures are correctly oriented.

We verify the accuracy of the local breath measure orientation algorithm by correlating the breath measures with the belt measurements. For all but one couch position of one patient we are able to correctly orient the breath measure, using this method. The patient on which our image based method fails is breathing erratically and the

method fails on a couch position where the correlation between the local breath measure and the belt measurements is 0.05.

#### 3.4.2.3 Aligning Local Breath Measures

A distance measure is defined between slabs as the squared distance between the local breath measures $f_1(x)$ and $f_2(y)$ of the two slabs after alignment by the transformation $\psi$.

$$D(x, y) = (f_1(x) - \psi(f_2(y)))^2 \tag{3.4}$$

The affine transformation which we use to align the coordinate spaces is a function with two parameters: the scaling value $a$ and the translation value $b$.

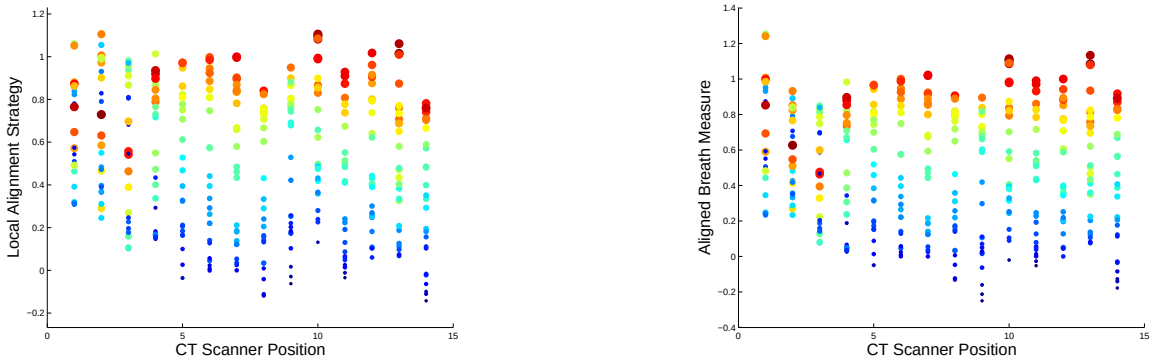
$$\psi : v \mapsto av + b \tag{3.5}$$

We require a cost function which measures the quality of an aligning transformation $\psi$ between the local breath measures of two neighboring couch positions. We create the cost function as the sum of the distance between the transformed local breath measure coordinates for each slab pair in the matching slabs list. The distances are normalized by the discontinuity measure to emphasize slab pairs which are good corresponding matches.

$$\begin{aligned} C(a, b) \;=\; & \sqrt{\sum_{x \in S_1} \frac{D(x, y_{x,1})}{\rho(x, y_{x,1})} + \frac{D(x, y_{x,2})}{\rho(x, y_{x,2})}} \\ & + \sqrt{\sum_{y \in S_2} \frac{D(x_{y,1}, y)}{\rho(x_{y,1}, y)} + \frac{D(x_{y,2}, y)}{\rho(x_{y,2}, y)}} \end{aligned} \tag{3.6}$$

To find the parameters $a$ and $b$ of the affine transformation which minimize the cost function we use sequential quadratic programming, a standard constrained optimization algorithm (implemented in Matlab as `fmincon`).

$$\arg\min_{a,b} C(a, b), \quad a > 0 \tag{3.7}$$




(a) (b)

Figure 3.7: (a) Global breath measure created by aligning two couch positions at a time, moving out from center. (b) Global breath measure created by simultaneously solving for the transformation of each local breath measure. This is the default strategy. The first three couch positions are above and at the top of the lung, where there is not very much motion and the Isomap based breath measure is not very correlated with the belt measurements (as shown in figure 3.8(a)).

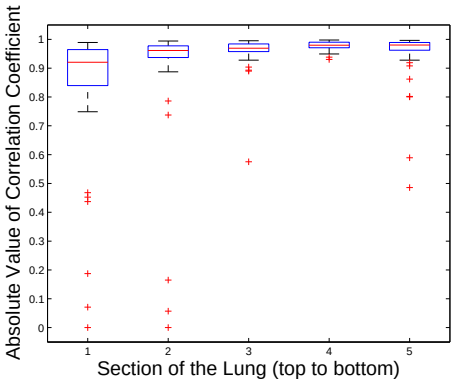
The constraint $a > 0$ is used so that the transformation cannot invert exhalation and inhalation for any local breath measure from that which was set explicitly in the last section. This constraint is necessary to avoid converging to local minima that are very far from optimal.

#### 3.4.2.4 Aligning all Couch Positions

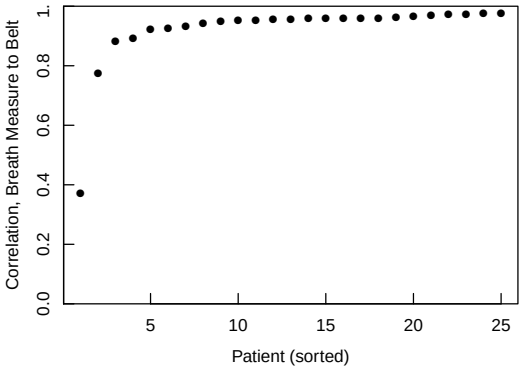
To create a global breath measure we must place all local breath measures into a single global coordinate space. We first describe a pairwise alignment strategy and then a more global strategy.

In the pairwise approach, we start with the middle couch position (position 7 in the figures) and rescale it to the range 0 to 1. Then, moving outward from this couch position, we align the local breath measures of each couch position to that of its neighbor closest to the center. This yields a global breath measure which consistently describes the breath phase for every slab in all couch positions as shown in figure 3.7.

Instead of the pairwise approach we can simultaneously solve for the $a$ and $b$ values of the transformation for each couch position. We can additionally constrain the $a$ values




(a) Correlation:
local breath measure and belt




(b) Correlation:
global breath measure and belt

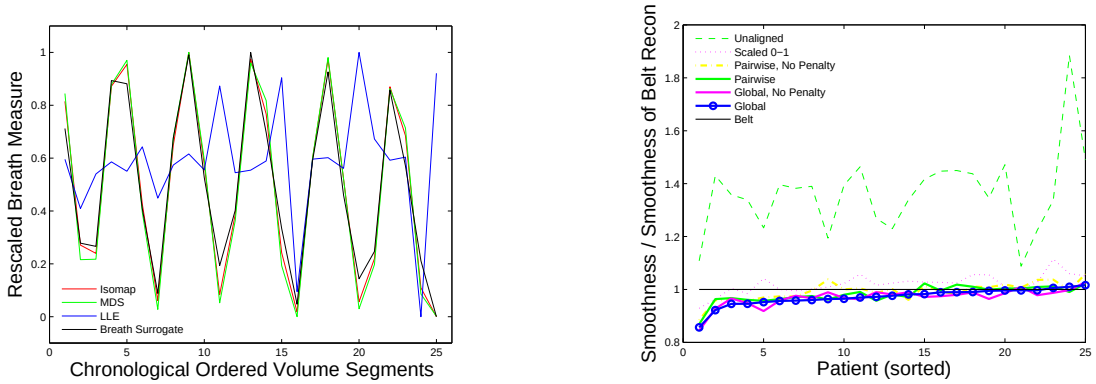
Figure 3.8: (a) Correlation of the local breath measure for different parts of the lung and the belt measurements for 25 patients. The lung is grouped into 5 sections from the top to the bottom. A boxplot of the absolute value of the correlation coefficient is shown for each section. (b) Correlation of computed global breath measures to external belt measurements for each patient. Patients are sorted by increasing correlation.

to have a mean of 1, ensuring that the transformation is not trivial. In section 3.4.3.3 we show that the best method is to solve for all transformations simultaneously and constrain the $a$ parameters of the transformations to have a mean of 1.

### 3.4.3 Breath Measure Algorithm Details

#### 3.4.3.1 Physiological Verification of Breath Measure

We compare the global breath measure against estimates of the lung motion captured by measuring the extension of a belt around the chest of the patient. Figure 3.8(a) shows the correlation of the breath measure with the belt measurements for all couch positions of all 25 patients grouped by location in the lung. The correlation is worse at the top of and above the lung where there is little breath related motion. Figure 3.8(b) shows the correlation of the global breath measure with the belt measurements for each patient. The patients are sorted by increasing correlation. These results show that the data driven method is very similar to the currently in use method based on the belt measurement.



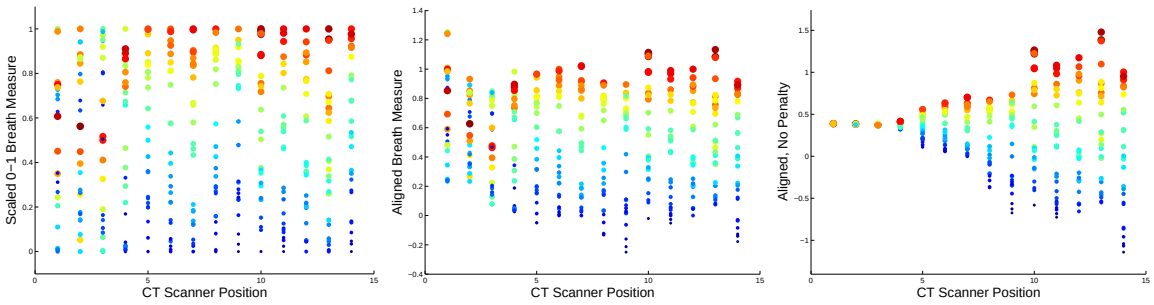

(a) Local Breath Measure at Diaphragm (b) Discontinuity Measure of Reconstructions

Figure 3.9: (a) Several different breath measures collected at a single scanner position near the diaphragm. This figure shows the similarity between the local breath measures acquired by the belt measurement, Isomap, and Multidimensional scaling. LLE does not produce a good breath measure. All breath measures are rescaled to the range [0,1]. (b) Discontinuity measure for reconstructions based on different breath measures. Our methods generally have values below one: this means that their reconstructions are smoother than reconstructions based on a belt measurement.

As another measure of the validity of the algorithm we compare the reconstructed volumes created using the Isomap based breath measure to volumes based on the belt measurements. The metric used for comparison is the discontinuity measure (equation 3.3) across each couch position boundary in the reconstructed volumes. Figure 3.9 shows the ratio of boundary discontinuity for a reconstruction based on a particular breath measure to the discontinuity of the reconstruction based on the belt measurements. Values below 1 mean that the given reconstruction algorithm is smoother than the reconstruction based on the belt measurements. Patients are sorted by the ratio for the aligned with penalty term reconstruction. These results show that using the Isomap parameterization of breathing produces smoother reconstructions than using belt measurements.

#### 3.4.3.2 Discussion of Local Breath Measures

The reconstruction algorithm hinges on the ability of Isomap to parameterize the breath phase of slabs. Several methods other than Isomap were investigated. Figure 3.9(a) shows the breath measures obtained for a single couch position near the




(a) High Penalty ($\alpha \to \infty$) (b) Optimal $\alpha$ (c) No Penalty ($\alpha = 0$)

Figure 3.10: A penalty term is necessary to maintain the range of the breath measure close to the range 0 to 1 in each couch position. However, this requires finding a value for the scaling constant $\alpha$ of this penalty term. (a) Large values of $\alpha$ force each local breath measure to have exactly the range 0 to 1. (b) Optimal $\alpha$ value, as determined by considering the variation in the output coordinates in figure 3.11(c), creates a global alignment with a consistent range. (c) Too small an $\alpha$ does not penalize the collapse of local breath measures onto a single point. Note that this happens even though we have forced the mean range of the parameters in each couch position to be 1.

diaphragm by different methods. Multidimensional scaling which is identical to Principal Component Analysis (PCA) in this setting and upon which Isomap is based has a very similar output to Isomap. The small number of slab samples means the neighborhoods used by Isomap in the graph cover a large area and most pairs of points are connected through short paths with only 1 or 2 edges. This causes the graph distances to degenerate into Euclidean distances and Isomap to be equivalent to MDS. Locally Linear Embedding (LLE) produces a breath measure significantly worse than the other methods [73]. This is probably due to the small number of samples at each couch position.

#### 3.4.3.3 Constraining Global Alignment to a Consistent Range

The alignment algorithm can produce differing breath measure ranges between the ends of the lung. For example, in figure 3.10(c), the couch positions near the top of the lung have a much smaller range in its breath space than couch positions in the lower part of the lung.

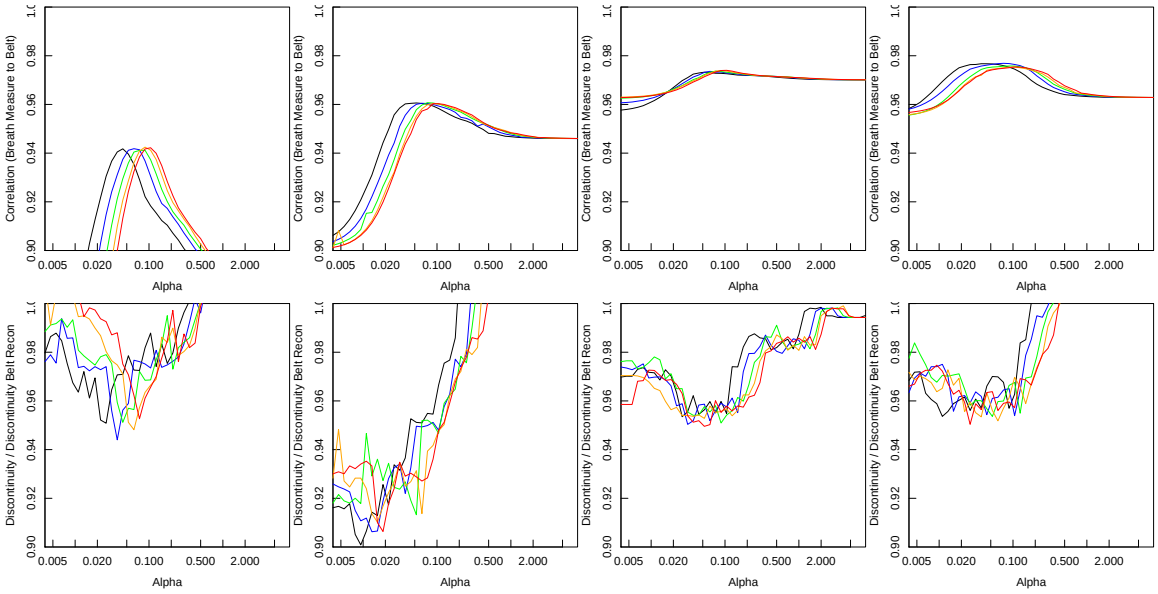
We extend the cost function in equation 3.6 to encourage a range of 0 to 1 for the breath measure by adding a penalty term.

$$C' = C + \alpha \left( (\max(f_i(S_i)) - 1)^2 + (\min(f_i(S_i)) - 0)^2 \right) \tag{3.8}$$

Setting the correct value for $\alpha$ is important for creating a global breath measure. Too large a value will force the range of each local breath measure to be exactly between 0 and 1. Too small a value will allow local breath measures to collapse to a single point. These problems are apparent in figure 3.10. To find a good value for $\alpha$, over all patients, we compare the global breath measure to the belt measurement, and pick the $\alpha$ which on average gives the breath measure with the highest correlation with the belt measurement. Each plot in figure 3.11(top) shows for a single patient the correlation between the breath measure and the belt measurements for varying values of $\alpha$ and varying number of matching slabs. This figure also shows the discontinuity measure computed across the couch position junctions in the reconstructed volume normalized by the same value for a reconstruction based on the belt measurement. Where the correlation between the breath measure and the belt measurements is high, we also see that the reconstruction based on the breath measure is smoother than one based on the belt measurements. Figure 3.12 shows the mean correlation and mean normalized discontinuity measure over all patients. The three methods considered are (a) aligning the couch positions in pairs moving outward from the middle couch position, (b) aligning all couch positions simultaneously without constraining the mean range in each couch position, and (c) aligning all couch positions simultaneously while constraining the mean range in each couch position to have a value of 1. The method with the highest average correlation of 0.943 over all patients constrains the mean range to have value 1, uses 3 matching slabs for the alignment, and uses a value of 0.0743 for $\alpha$. These are the parameter values we use unless specified otherwise.

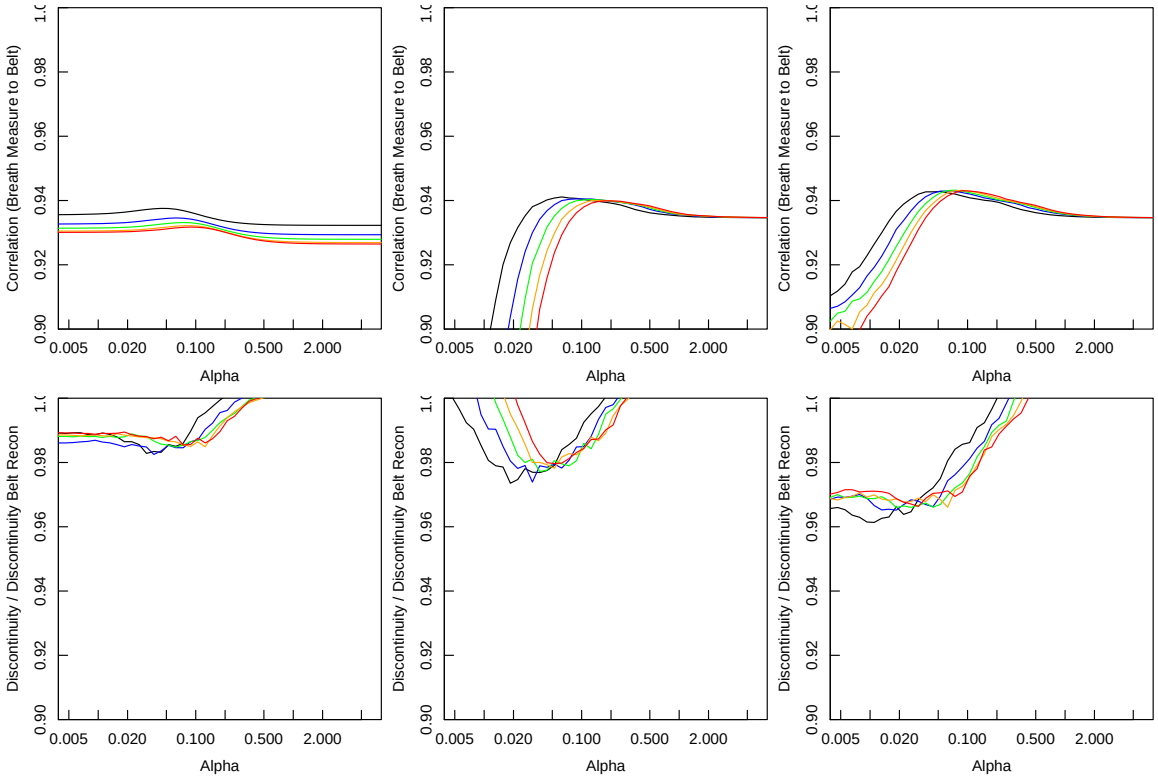
### 3.4.4 4D CT Results with Data Driven Breath Measure

We show the 4D CT lung models that result from the algorithms presented in this section. Figure 3.13 shows various different breath measures for 4 patients. The first row is the external breath marker provided by the belt and is used as a basis for comparison, the size and hue of the dots in each figure are proportional to these values.



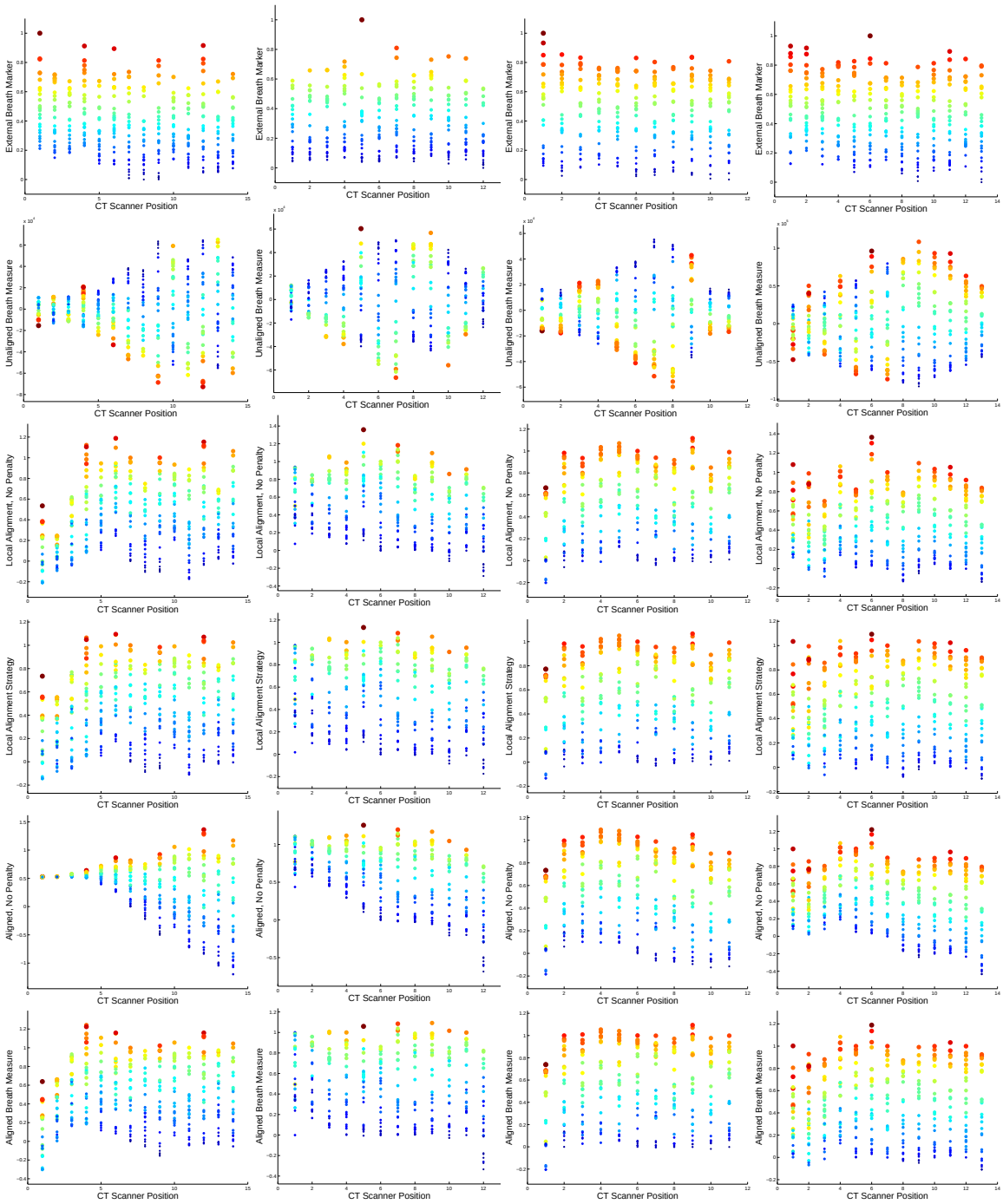

(a) Patient 38 (b) Patient 54 (c) Patient 56 (d) Patient 58

Figure 3.11: Top: The correlation between the breath measure for a particular value of $\alpha$ and the belt measurement for 4 different patients. Higher values indicate that the breath measure and the belt measurements agree. There are 5 different lines each corresponding to a different number of matching slabs (found by the discontinuity measure) used in creating the alignment. Black is 1 matching slab, blue 2, green 3, orange 4, and red 5. The x-axis is displayed in a log scale. Bottom: For the same 4 patients, we show the discontinuity measure between all junctions in the reconstruction normalized by the same value for the reconstruction based on the belt measurements. Lower values indicate that the reconstruction based on the breath measure is smoother than that based on the belt measurements.



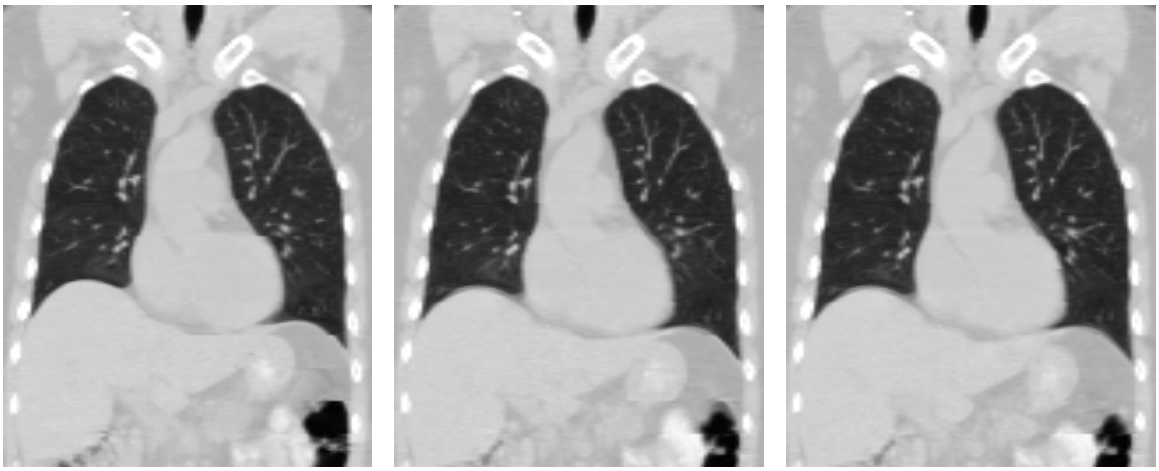

(a) Pairwise Alignment (b) Global Align (unconstrained) (c) Global Alignment

Figure 3.12: Three different methods of creating alignments are compared (each in a column) using two different methods (shown in the rows). In each plot, the five different lines correspond to different numbers of matching slabs (Black is 1 matching slab, blue 2, green 3, orange 4, and red 5). Top: The correlation between the breath measure and the belt measurement is plotted. Bottom: The discontinuity measure between all junctions in the reconstruction normalized by the same value for the reconstruction based on the belt measurements. Lower values indicate that the reconstruction based on the breath measure is smoother than that based on the belt measurements. (a) Couch positions are aligned in pairs moving outward from the center. (b) All couch positions are simultaneously aligned. (c) All couch positions are simultaneously aligned and the mean range in each couch position is constrained to have the value 1. The best combination (in the top row) with an average correlation of 0.943 is found in plot (c) with alpha value 0.0743 and 3 matching slabs used.

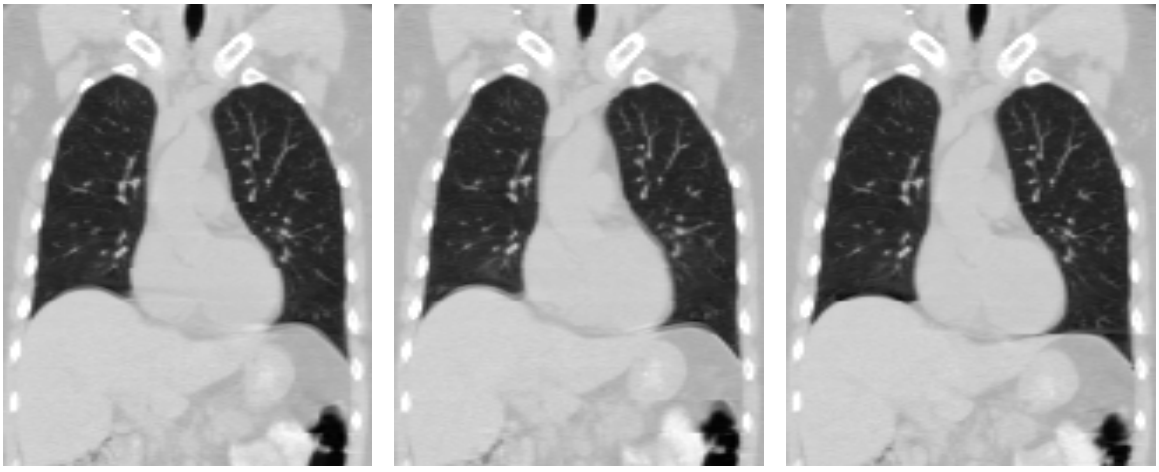



Patient 38 Patient 54 Patient 56 Patient 58

Figure 3.13: Different breath measures for several patients. Each row is a different breath measure algorithm; each column is a different patient.

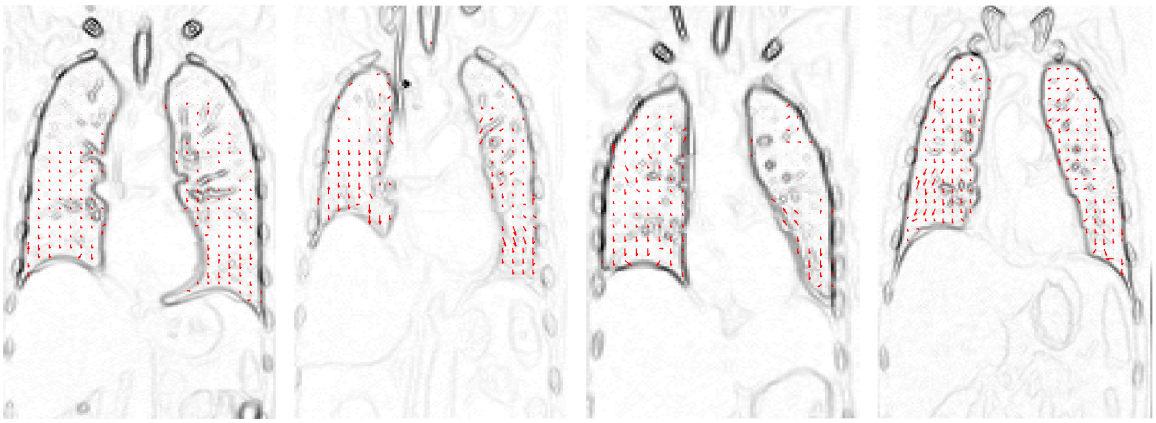


Unaligned Scaled 0-1 Local Align

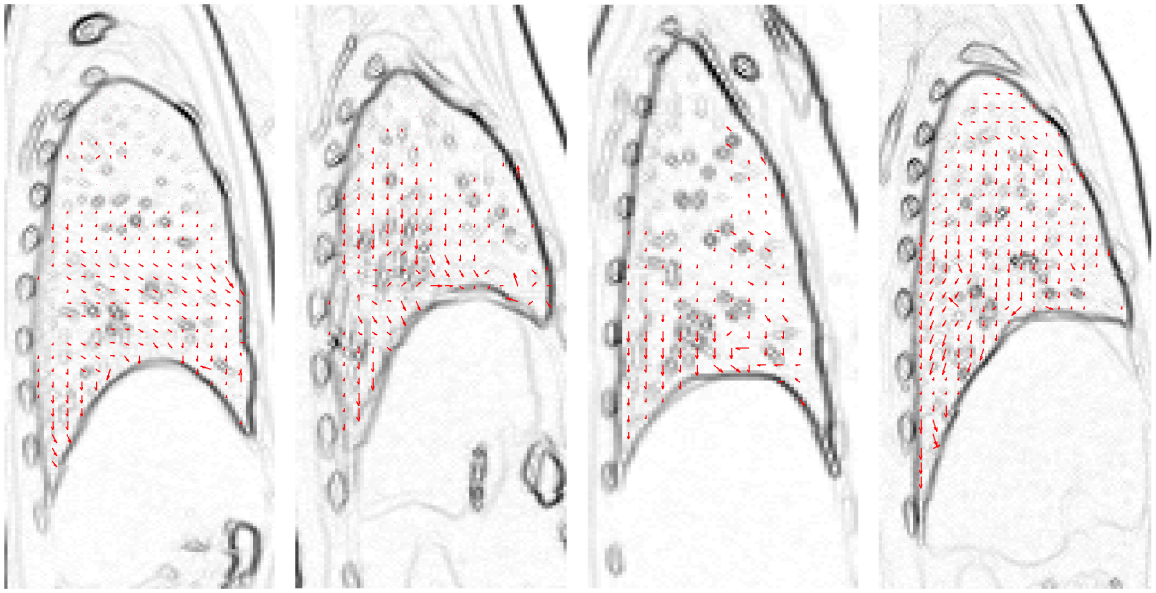


No Penalty With Penalty Belt

Figure 3.14: Coronal cross-section image of the reconstructed lung of patient 38. Each image is at a breath phase relatively close to full inhalation (0.8, where 0 is exhaled and 1 is inhaled). Note the artifacts in the unaligned reconstruction and that our final solution "with penalty" is essentially indistinguishable from the method that requires the external belt measurements.

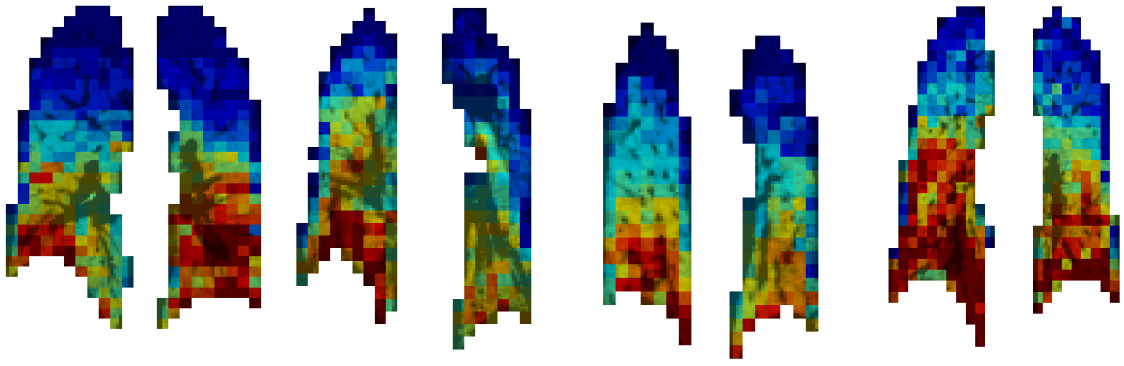


Patient 38 Patient 54 Patient 56 Patient 58

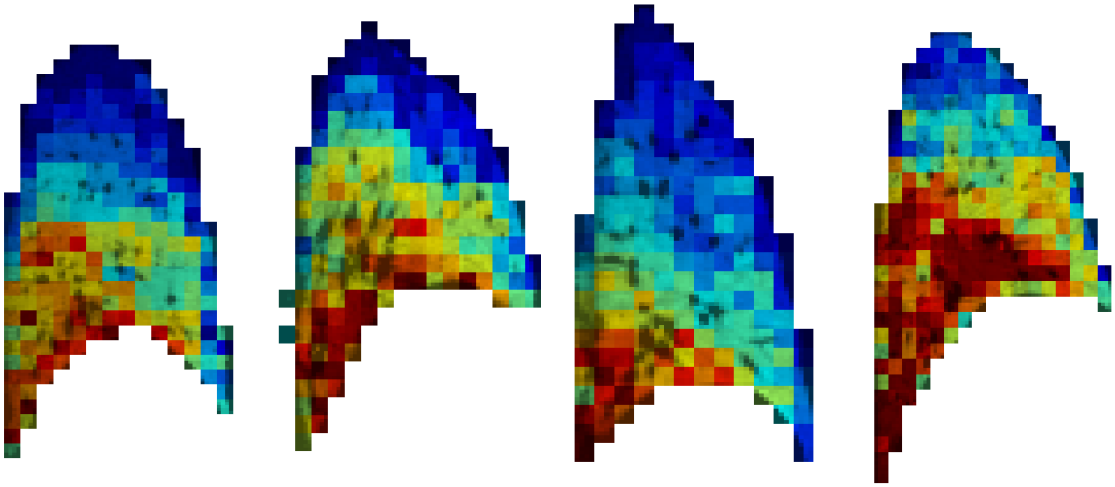


Patient 38 Patient 54 Patient 56 Patient 58

Figure 3.15: Movement vectors for several patients. Top: coronal cross-sections. Bottom: sagittal cross-sections.



Patient 38 Patient 54 Patient 56 Patient 58



Patient 38 Patient 54 Patient 56 Patient 58

Figure 3.16: The magnitude of motion in patient lungs. Red sections move more than blue sections. Top: coronal cross-sections of patients. Bottom: sagittal cross-sections of patients.

The second row shows the raw Isomap generated coordinates for each couch position before any alignment is performed. The orientation and scale of the local breath measure at each couch position is arbitrary and unrelated to any of the other couch positions. However, these raw Isomap coordinates form the basis for all the other global breath measures. The third row shows the pairwise alignment strategy without a penalty term. The breath measure in each couch position is oriented properly and aligned to its neighbors; however, there is no effort to maintain the breath measure within a particular range and the breath measure tends to drift or shrink from one end of the lung to the other. The fourth row shows the pairwise alignment strategy with penalty term of $\alpha = 0.0442$ and 1 matching slabs, which is the best strategy as determined in figure 3.12(a). The added penalty term maintains the range near 0-1. However, the pairwise alignment strategy does not find as good an alignment as the global alignment strategy. The fifth row is the global alignment strategy without a penalty term. The global alignment strategy is able to more effectively minimize the cost function in equation 3.6. Since this equation does not include a penalty term to maintain the range 0-1, the drift and scale collapse problem is very pronounced in some patients. Finally, the sixth row is the global alignment strategy with a penalty term of $\alpha = 0.0743$ and 3 matching slabs, which is the optimal alignment strategy as determined in figure 3.12(c) The penalty term maintain the range near 0-1 and the global alignment strategy allows the optimization algorithm to more effectively find the cost function minimum. These are the coordinates which will be used in the next part of the analysis.

Lung reconstructions at any breath phase can be generated using a breath measure. Figure 3.14 shows coronal cross-sections of the reconstructed lung generated for a mostly inhaled breath phase (0.8, where 0 is exhaled and 1 is inhaled). Several different breath measures are used to generate these images, the best data driven method is labeled "With Penalty" and for comparison is placed next to the reconstruction created with the belt measurements. Although there are differences between the images, since we are only showing a single coronal cross-section of the 3D volume, it is very difficult to visually determine which image is better. Additionally, although it is instructional to look at reconstructions, for evaluation, a quantitative method such as that in figure 3.9(b) should be used. The strength of a reconstruction should not be judged solely on discontinuities along the edge of the lung, since the discontinuities

inside the lung, although more difficult to discern are more important to an understanding of tissue motion. In particular, the heart in the lower right-middle section of each image in figure 3.14 frequently causes discontinuities, which are difficult to avoid regardless of the method used in the reconstruction. For more context on what is causing these discontinuities recall that a reconstruction is created by picking the slab with breath phase closest to a target value, in this case a value of 0.8. Looking at the corresponding breath measure space for these reconstructions and following along on a line of value 0.8 the reason for many errors becomes clear, since the line goes above the entire range for some couch positions (figure 3.13 column 1, row 1, 5, and 6). In the "Belt" reconstruction there are several couch positions with discontinuities on both their top and bottom portion (for example the couch position containing the fourth rib from the top). In the "No Penalty" reconstruction the entire top half of the lung is in a full inhaled state. In the "With Penalty" reconstruction there are some discontinuities near the top of the lung (third rib from the top) where the breath measure drifts from full inhalation to near inhalation. In general, our reconstructions are comparable, and frequently a little smoother, than reconstructions based on a belt measurement. Additionally, creating a reconstruction similar to the belt based reconstruction without the use of any external measurements is a significant contribution with clinical applications.

Figure 3.15 shows images of tissue motion created using the same technique as in section 3.3.4 except using the global breath measure instead of the belt measurements. Tissue motion is tracked through 8 lung reconstructions generated by the global breath measure. Both coronal and sagittal cross-sections of the lung are shown for each patient. The template matching algorithm functions a little better when applied to the smoother reconstructions produced by the global breath measure; however, the problem of ambiguity and problems tracking tissue when discontinuities are present remain. Since the method presented in the next section addresses these problems we refrain from an evaluation of the template tracking method.

Figure 3.16 shows the magnitude of motion computed by following motion vectors through the entire breath phase as is described in section 3.3.5. Sections of the lung which have more motion are shown in red while parts of the lung with little motion are in blue. Both coronal and sagittal cross-sections of the lung are shown for each patient. From these images it is clear that tissue in the lower portion of the lung moves

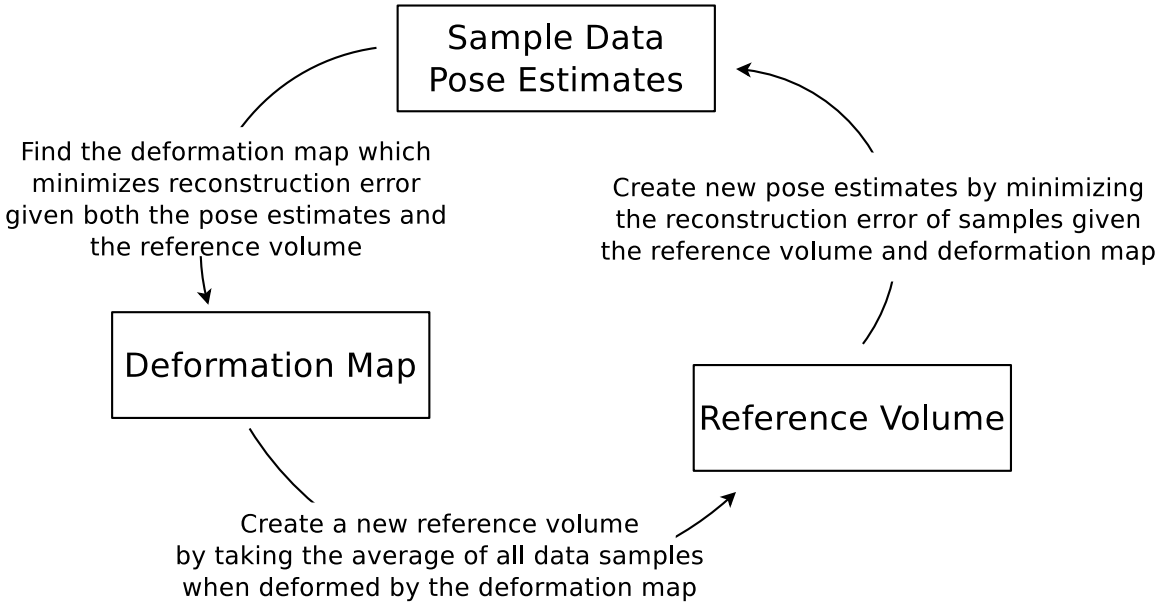



Figure 3.17: The optimization procedure for creating a deformation map and reference lung based model.

significantly more than tissue in the upper lung. These motion maps are useful in determining the margins which must be placed around a tumor when computing a radiation treatment plan.

## 3.5 Free Form Deformation Based Modeling of Tissue Volumes

We now focus on the alternative approach of modeling the lung tissue movement explicitly with a free form deformation model. This approach allows us to better leverage redundant information and allow for missing data at particular breath phases. Figure 3.17 shows the three parts involved in this approach.

We broaden our problem formulation to include problems with more than one cause of variation. Medical images of moving tissue can be represented as a function $I(\vec{x}, \vec{\theta})$ which captures the intensity of tissue in the image at all spatial locations $\vec{x}$ and poses $\vec{\theta}$. For the 4D CT lung the vector $\vec{x}$ represents the three spatial dimensions of volume and the pose dimension $\vec{\theta}$ is a one dimensional space representing breath phase (or

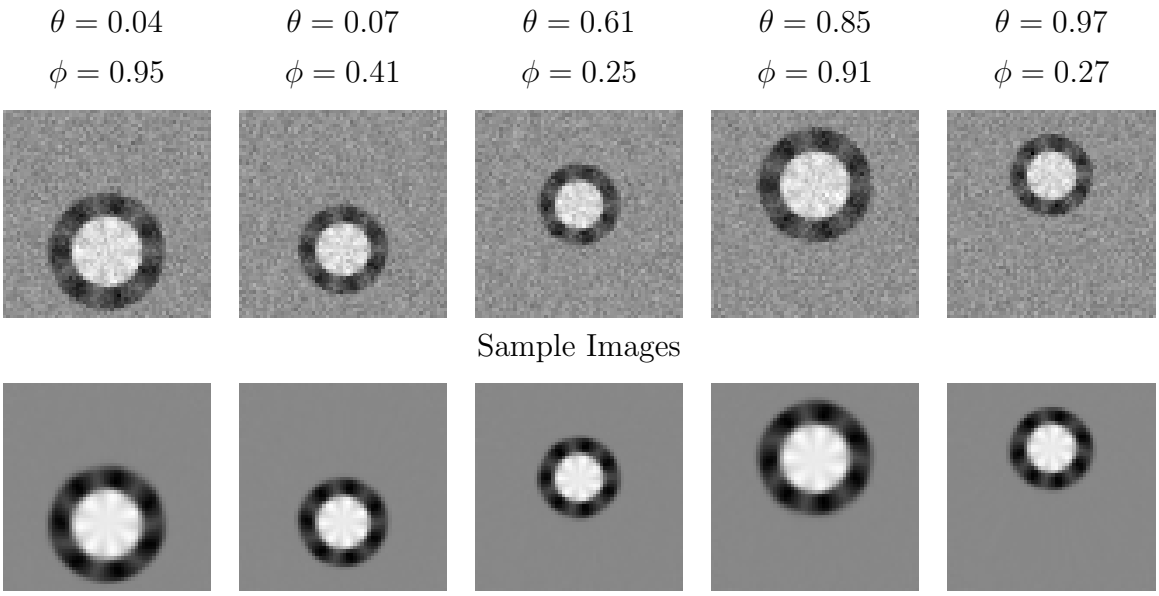



Figure 3.18: Image samples and their reconstructions. The reconstructions contain less noise, since they combine the data from many samples.

lung air volume). Given data samples $S_i(\vec{x})$ with pose $\vec{\theta_i}$ we seek a function $I(\vec{x}, \vec{\theta})$ such that $I(\vec{x}, \vec{\theta_i}) = S_i(\vec{x})$. We create a model $I(\vec{x}, \vec{\theta})$ which consists of a reference lung model and a deformation map.

The algorithm presented in this section is applicable to scenarios other than 4D CT. As examples, we will use 4D CT data, heart/lung MRI data, and synthetic heart/lung data. The 4D CT data has 3 spatial dimensions and 1 breath space dimension, the heart/lung MRI data and synthetic heart/lung data each have 2 spatial dimensions, 1 breath and 1 heartbeat dimension. The synthetic data shown in figure 3.18 will be used as the main example in the presentation of the algorithm.

### 3.5.1 Deformation Model

To constrain the data volume $I$ we assume that it is created from an underlying reference volume $I_r$ which accounts for the appearance of the structure in all spatial dimensions and a set of free form deformations $\vec{f}(\vec{x}, \vec{\theta})$ parameterized by pose $\vec{\theta}$ which account for the motion of the structure during physiological activity. We use cubic

B-Splines as our free form deformation for simplicity of computation and differentiability and for the finite support of its basis functions. For simplicity, we present our algorithm using an example with just 1 spatial and 1 pose dimension. A cubic B-Spline deformation is a continuous piecewise polynomial function, with continuous first and second derivatives. Each span of the B-Spline can be written as a weighted sum of separable basis functions.

$$\vec{f}(x,\theta) = \sum_{a=1}^{4}\sum_{b=1}^{4} A_a(x)B_b(\theta)\vec{w}_{ab} \tag{3.9}$$

$A_a$ and $B_b$ are cubic basis functions with joint weight $\vec{w}_{ab}$. These weights can be thought of as a grid of control points which describe the deformation map. The dimensionality of the weight vector $\vec{w}_{ab}$ equals the number of spatial dimensions in the model. The deformation takes as input the spatial position and pose of the structure, and outputs the deformed position of the structure in a reference coordinate system. The deformed image $I_d$ at a pose $\vec{\theta}$ is calculated by indexing the reference image $I_r$ with the deformation map.

$$I_d(\vec{x},\vec{\theta}) = I_r(\vec{f}(\vec{x},\vec{\theta})) \tag{3.10}$$

### 3.5.2 Deformation Model Optimization

The model is defined by three sets of parameters which correspond to the three kinds of inference problem.

1. The values of the voxels in the reference volume $I_r$.
2. The parameters of the deformation map $\vec{w}_{\vec{a}}$.
3. The pose estimates $\vec{\theta}_i$ for each sample.

The cost of the model is defined as the reconstruction error of the samples.

$$C = \sum_{S_i \in \mathbb{S}} \int (I_d(\vec{x},\vec{\theta}_i) - S_i(\vec{x}))^2 \, dx \tag{3.11}$$

The integral is defined over all the pixels of the sample image $S_i$, which may only be part of the range of the model.

We analytically compute the derivative of this cost function with respect to both the weights $\vec{w}_{\vec{a}}$ and poses $\vec{\theta}_i$ allowing the use of an efficient solver to compute the cost function minimum. A standard non-linear solver L-BFGS-B which uses a Quasi-Newton approach is used to solve for the minimum of the cost function [17]. The variables over which the solver optimizes are both the B-Spline weights and the pose estimates. In each iteration a new reference image is solved for directly, given the current set of pose estimates and deformation map.

#### 3.5.2.1 Basis Weight Derivatives

For clarity we present the following equations for the case with exactly one spatial dimension and one pose dimension. The gradient of the cost function with respect to a basis function weight can be computed analytically.

$$\begin{aligned} \frac{\delta}{\delta w_{\alpha\beta}} C &= \frac{\delta}{\delta w_{\alpha\beta}} \sum_{S_i \in \mathbb{S}} \int (I_d(x, \theta_i) - S_i(x))^2 \, dx \\ &= 2 \sum_{S_i \in \mathbb{S}} \int (I_d(x, \theta_i) - S_i(x)) \frac{\delta}{\delta w_{\alpha\beta}} I_d(x, \theta_i) \, dx \end{aligned} \tag{3.12}$$

The derivative of the deformed image with respect to the weight of a basis function can be calculated using the chain rule on equation 3.10.

$$\frac{\delta}{\delta w_{\alpha\beta}} I_d(x, \theta_i) = \frac{\delta}{\delta w_{\alpha\beta}} I_r(f(x, \theta)) = \frac{\delta I_r(f(x, \theta))}{\delta f(x, \theta)} \frac{\delta}{\delta w_{\alpha\beta}} f(x, \theta) \tag{3.13}$$

The generalized form of the chain rule is required when the model has more than one spatial dimension; however, this value is still easily computed. The first term of the product can be calculated directly as an image derivative of the deformed image. The second part can be calculated from the B-Spline using equation 3.9.

$$\frac{\delta}{\delta w_{\alpha\beta}} f(x, \theta) = \frac{\delta}{\delta w_{\alpha\beta}} \sum_{a=1}^{4} \sum_{b=1}^{4} A_a(x) B_b(\theta) w_{ab} = A_\alpha(x) B_\beta(\theta) \tag{3.14}$$

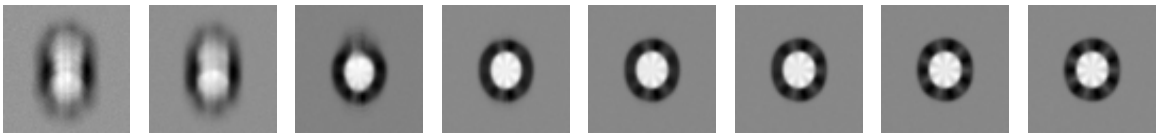


Figure 3.19: Reference images of synthetic data in iteration 1, 43, 143, 243, 343, 443, 543, and convergence at 643.

This just picks the basis function which corresponds to the weight $w_{\alpha\beta}$.

#### 3.5.2.2 Pose Estimate Derivatives

The gradient of the cost function with respect to the pose $\theta_k$ of a sample can also be calculated analytically. Similarly to equation 3.12, the derivative of the cost function can be calculated; however, only one sample will be non-zero.

$$\frac{\delta}{\delta\theta_k}C = 2\int (I_d(x,\theta_k) - S_k(x))\frac{\delta}{\delta\theta_k}I_d(x,\theta_k)\,dx \tag{3.15}$$

The partial derivative of the B-Spline function with respect to $\theta_k$ requires taking the derivative of some of the basis functions of the B-Spline.

$$\frac{\delta}{\delta\theta_k}f(x,\theta_k) = \sum_{a=1}^{4}\sum_{b=1}^{4} A_a(x)B_b'(\theta_k)w_{ab} \tag{3.16}$$

$B_b'$ is the derivative of $B_b$ and the fact that $B_b'$ is easily computed motivated our choice of B-Splines to model the deformation.

#### 3.5.2.3 Computing Reference Images

The derivative of the cost function with respect to the reference image is not easy to compute analytically.

$$\frac{\delta}{\delta I_r(x_0)}C = \frac{\delta}{\delta I_r(x_0)}\sum_{S_i\in\mathbb{S}}\int (I_d(\vec{x},\vec{\theta}_i) - S_i(\vec{x}))^2\,dx \tag{3.17}$$

This quantity is dependent on both the method of computing the deformed image from the reference image and the method of approximating the integral over the

discrete space of the deformed image. Additionally, having the solver automatically compute each pixel of the reference image would dramatically increase the number of variables, making the problem unmanageable. Instead, in each iteration we compute a new reference image directly from the deformation map and pose estimates.

A reference image can be computed given a deformation map and pose estimates. Each sample image can be deformed into the coordinate system of the reference image, using the inverse of the B-Spline deformation. This reference image is optimal with respect to the sum of squared difference between the deformed images and the samples, given the deformation and pose estimates. Figure 3.19 shows the evolution of the reference image as the optimization iterates.

#### 3.5.2.4 Initial Conditions

The identity transformation is used as an initial guess for the deformation. The initial pose estimates can be measured directly using a physical device, computed from imaging data using standard manifold learning techniques or for lungs using the earlier presented breath measure algorithm. Assuming the identity transform as an initial deformation leads to an initial reference image which is the average of all the samples.

#### 3.5.2.5 Anchoring the Reference Image

As described so far, the optimization problem is under-constrained, for example, a shift of the reference image is equivalent to a shift of the deformation map. Furthermore, there is no guarantee that the reference image even resembles the tissue. This problem can be solved by constraining the reference image to be equal to the tissue volume at a particular pose which we call the central pose. As the optimization proceeds, we add a term which penalizes deviation from the identity map of the deformation map at the central pose. A large scaling factor on this penalty term can lead to slow convergence and may adversely effect pose estimates if they are allowed to change during optimization. Figure 3.20 shows a sequence of final reference images

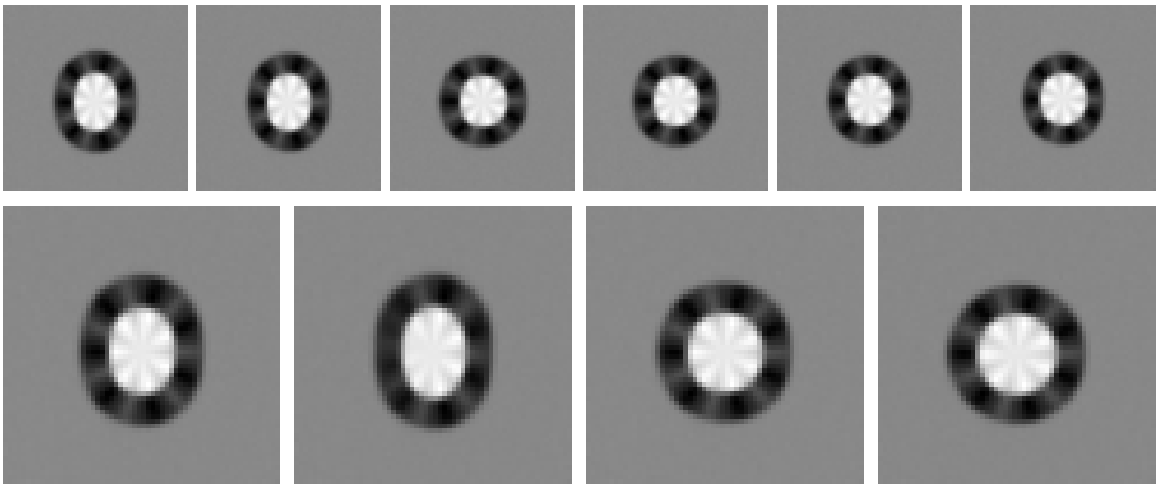


Figure 3.20: Final reference images as the deformation is constrained to be closer to the identity at the central pose. The top row uses the penalty approach which penalizes deviation from the identity map of the deformation map at the central pose. The penalty term increases from a low value of 0.125 to a value of 512 in multiples of 8. Bottom: The second row shows results for the faster method of directly fixing the central control points in the optimization algorithm. From left to right an increasing number of control points are fixed, starting with none on the left and all control points in the 3 by 3 grid around the central pose on the right.

for increasing penalty terms; notice that the reference images become rounder and more like the sample images as the level of constraint on the deformation is increased.

Unfortunately, with increasing penalty, the algorithm takes longer to converge. Another method of constraining the deformation map is to fix some of the control points of the B-Spline. This removes constraints from the solver which speeds up computation. Unfortunately, since B-Splines do not interpolate control points, this method does not force the deformation map to be the identity deformation at any particular point. Figure 3.20 (bottom) shows results for this method.

### 3.5.3 Results for Free Form Deformation Model

In this section, we show results of our algorithm for three data sets: a synthetic data set, a 4D CT lung data set, and a 2D heart MRI data set.

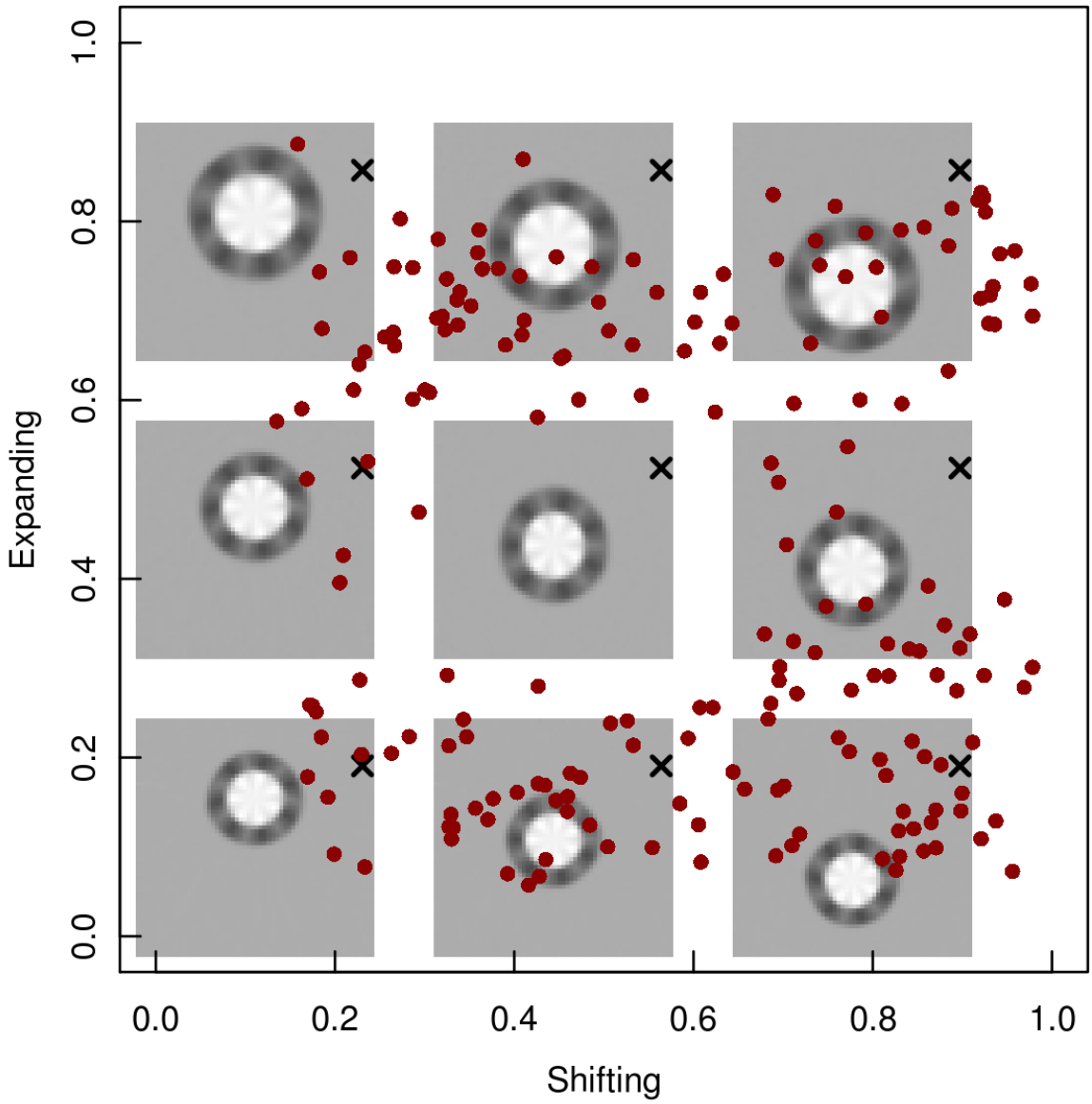
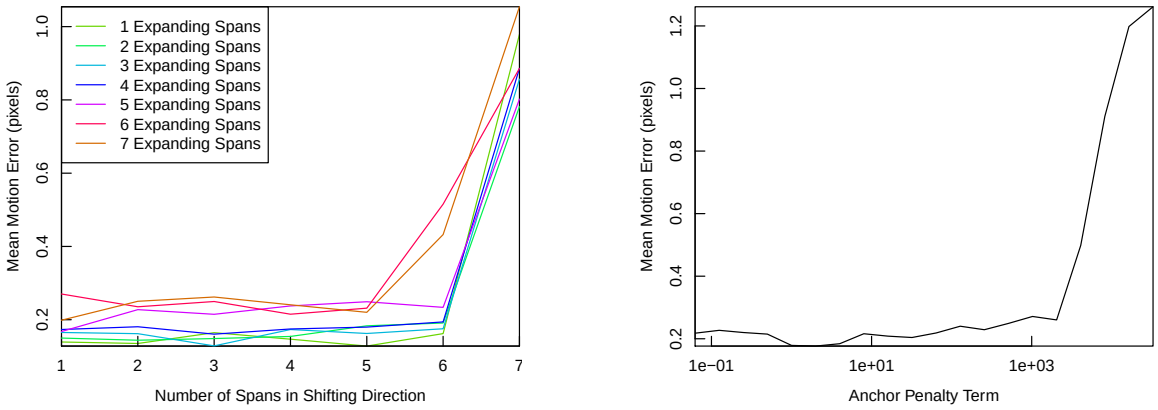



Figure 3.21: Pose estimates (red dots) for synthetic data. Reconstructed images of specific poses marked with a × are shown.




(a) Motion Error: Number of Spans

(b) Motion Error: Anchor Penalty

Figure 3.22: The mean pixel distance error between the model computed motion and the ground truth motion between each sample. (a) A variable number of B-Spline spans over the $\theta$ (shifting) and $\phi$ (expanding) parameter dimensions is used. On the x-axis, the number of spans in the $\theta$ (shifting) parameter direction is shown, and the different colored lines represent different numbers of spans in the $\phi$ (expanding) direction. The lowest error occurs when 5 spans are used for $\theta$ (shifting) and 1 span for $\phi$ (expanding). The error for 3 spans for each parameter dimension is very close to this error value. In general, a small number of spans is able to model this dataset well. (b) The same mean pixel distance error between the model and ground truth is shown when the anchor penalty is varied between a value of 0 and 32768 (x-axis is in a log scale).

#### 3.5.3.1 Analysis of Synthetic Data

We create a synthetic dataset of the heart consisting of two concentric slightly textured circles which deform in two ways: shifting, which simulates the effects of breathing on the heart, and expansion, which simulates a heartbeat. Each of the 200 sample images is 60 by 60 pixels and has Gaussian noise added. The Gaussian noise has a standard deviation which is 5% of the total intensity range of the object. Figure 3.18 shows some sample images and their reconstructions. Figure 3.21 displays the data volume of this synthetic data. Each black dot represents the location in parameter space of one of the 2D sample images used as input to the algorithm. At the poses marked with $\times$ we draw the cross section of the data volume $I(*, *, \theta, \phi)$ computed by our algorithm. The generating size and positions were chosen randomly and were not available to the algorithm. Instead, as in real applications, the initial poses were estimated using Isomap. Since the data was sampled uniformly and independently in both $\theta$ and $\phi$ we use Independent Component Analysis to create more meaningful axes.

For a model with B-Spline control points placed on a 10 pixel grid (6 spans in both $x$ and $y$ coordinates) with 5 spans over the shifting ($\theta$) parameter and 5 spans over the expansion ($\phi$) parameter for a total of 10368 parameters, optimization took about 1 minutes on an Intel Core 2 Q9550.

We evaluate the effect of changing the number of spans used in the $\theta$ and $\phi$ directions. Figure 3.22(a) shows the average difference between the computed motion and the ground truth motion (computed within the object) between every pair of samples. The number of spans in the $\theta$ and $\phi$ directions are varied between 1 and 7. The number of spans in the $\theta$ direction is shown along the x-axis and the number of spans in the $\phi$ direction is shown as different colored lines. The lowest error occurs when 5 spans are used for $\theta$ (shifting) and 1 span for $\phi$ (expanding). The error for 3 spans for each parameter dimension is very close to this error value. In fact, there is little difference in the error when 6 or less spans are used in the $\theta$ (shifting) direction and 3 or less spans are used in the $\phi$ (expanding) direction. When 7 spans are used in the $\theta$ (shifting) direction the algorithm does not converge within the maximum number of iterations (1000 iterations) and hence the error is much higher. In general, a small

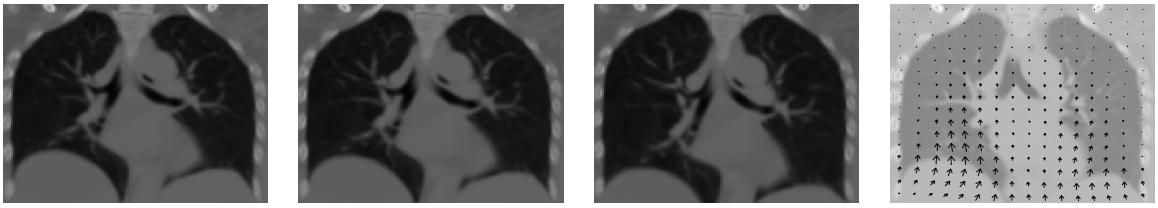


Figure 3.23: Lung reconstructions arranged from exhalation to inhalation. Right: vectors of tissue movement during breathing.

number of spans is able to model this dataset well, since the motion and deformation of the object is very simple.

We additionally evaluate the effect of changing the anchor penalty in figure 3.22(b). With too large an anchor penalty the algorithm takes too long to converge and is aborted, this significantly increases the motion estimation error. However, apart from this effect changing the anchor penalty does not change the motion estimation very much, even though it dramatically changes the shape of the reference image itself as seen in figure 3.20.

#### 3.5.3.2 Analysis of 4D CT lung

We apply our free form deformation model to the 4D CT lung dataset described in section 3.3.1. This method allows us to better model the lung tissue by leveraging the fact that the same tissue is being repeatedly sampled under various deformations.

Initial estimates of the breath phase (lung volume) were provided by a belt measuring patient chest circumference. Control points were placed every 8 voxels in all spatial dimensions, and 4 spans were used to model breathing, producing a total of 57,330 B-Spline parameters. The reference image has dimension 56 by 80 by 152 (680,960 voxels). And each of 250 samples has a single breath parameter. So 57,580 parameters are explicitly optimized for in the solver. Optimization took about 3 hours on an AMD Athlon 64 3800. Figure 3.23 shows a coronal cross section of the lung at three stages during exhalation and a vector map showing tissue motion during breathing.

To measure the quality of the deformation model we determine how well the model is able to register slabs with similar pose estimates. For each of the 250 sample

slabs we deform the sample slab with pose estimate most similar to it using the deformation map to account for motion due to breathing. The average voxel-wise difference between the sample and its deformed nearest neighbor is found to be 15.8 Hounsfield Units (HU) with a standard deviation of 4.0 HU. Hounsfield Units are a standard unit of X-ray absorption with a value of -1000 through air and 0 through water (fat has a value of approximately -120, and muscle approximately 40). In comparison, a direct difference between the sample and its nearest neighbor (without deformation) is 16.2 HU with standard deviation 4.3 HU. Although the improvement is small, a two tailed paired t-test over all 250 samples shows that the difference in mean between the two distributions is significant (p-value $2.1 \cdot 10^{-4}$).

#### 3.5.3.3 Analysis of MRI Heart/Lung

Data was captured as 193 frames of an un-gated cardiac MR during free breathing conditions. Initial pose estimates were computed using Isomap with a modification for embedding onto a cylinder [70]. Results are shown in figure 3.26. The vertical translation of the heart due to breathing is visible along the x-axis, while the cyclic heartbeat is visible along the y-axis. The deformation field along the arrows marked 1 and 2 are shown on the side.

To validate the model we remove 4 samples before optimization (shown in figure 3.24 as red numbers) and evaluate at the poses corresponding to these points. Figure 3.25 shows these samples, their reconstructions, and the estimated motion fields with respect to breathing and heartbeat. Although the images are accurately reconstructed, since the images vary in part due to blood flow that appears in some parts of the heartbeat cycle, the deformation field does not always correspond to tissue motion.

For further validation, we run a set of experiments in which each sample is in turn left out of the optimization. The mean reconstruction error of the left out sample is on average only 9% higher than the mean reconstruction error of samples left in. For context, the standard deviation of the normalized reconstruction error for samples that were left in the model was 39%.

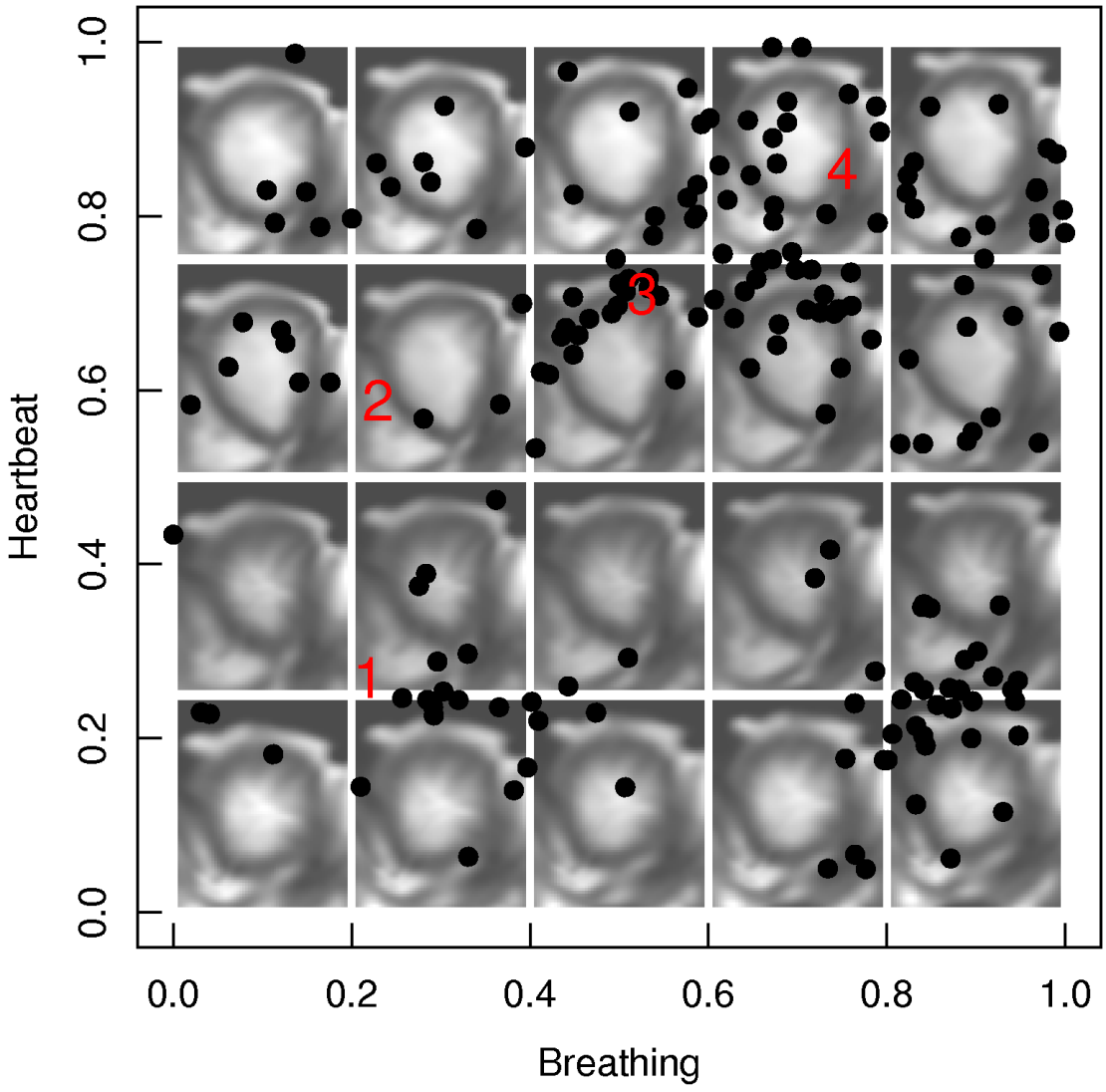



Figure 3.24: Pose estimates overlaid on reconstructed images showing a regular grid of poses. Red numbers indicate the location of samples which were excluded from the model and are shown in figure 3.25.

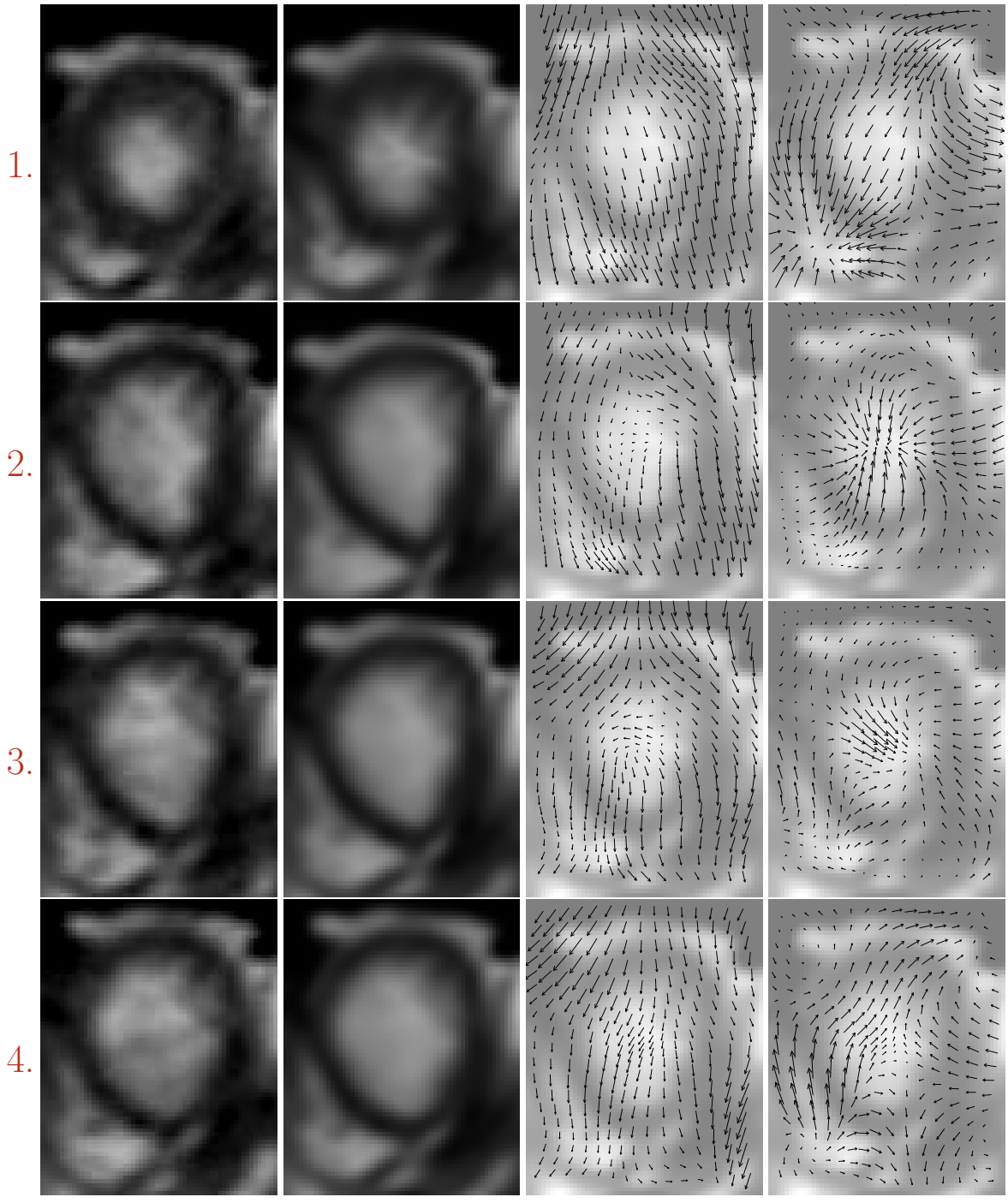



Figure 3.25: (a) Sample excluded from the deformation computation. (b) Reconstruction at the sample's pose. (c) and (d) Motion estimate due to breathing and heartbeat respectively.

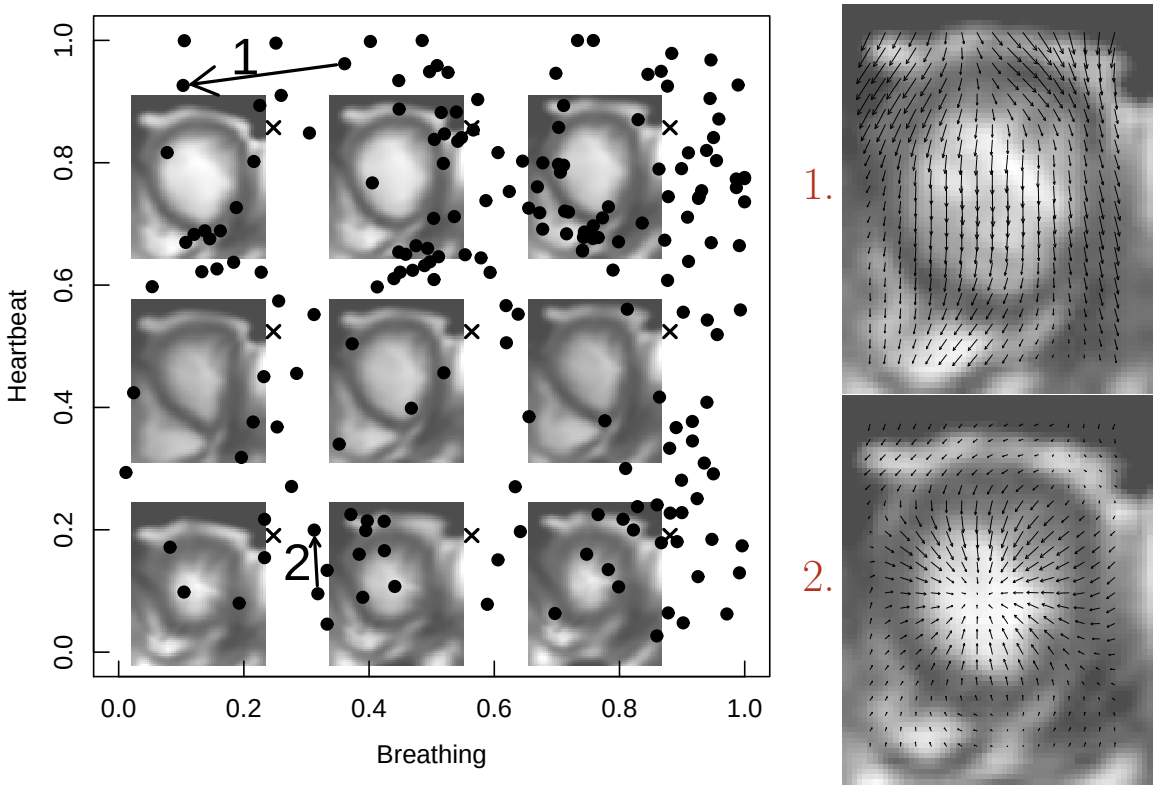



Figure 3.26: (Left) Large dots show the pose estimates of the MRI heart/lung dataset. Reconstructed images are shown at points marked with ×. The arrow 1 and 2 correspond to pose changes along the breathing and heartbeat axes respectively and are shown on the right side.

## 3.6 Conclusion

We provide a framework for modeling moving tissue in medical images. We presented two main contributions. The first contribution is a breath measure in 4D CT lung models which is completely data driven and does not require an external belt measurement. This breath measure can be used instead of the belt measurement to create traditional 4D CT lung models, even in cases where there was no belt measurement or the belt measurement is inaccurate. Our second contribution was to create a general framework for modeling medical tissue motion using a reference volume and a deformation map. The reference volume represents the underlying structure of the tissue and the deformation map, which is parameterized by the ways in which the tissue moves (*i.e.* breathing, and heartbeat), represents the motion of the tissue during physiological activity. This method simultaneously optimizes three parts: (1) the underlying undeformed appearance of the data volume, (2) the arbitrary deformations of tissue during physiological activity, and (3) the pose of data samples.

# Chapter 4

# Partially Unsupervised Manifold Learning

## Abstract

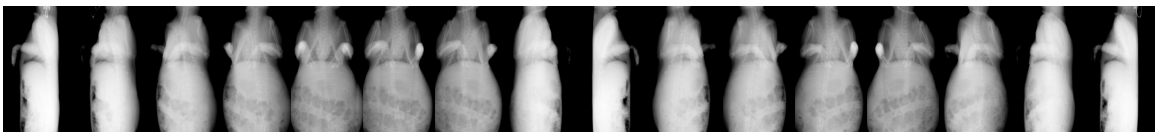
*We consider a special case of manifold learning where, for each point, one (of its few) low dimensional coordinates is known. The goal of our method will be to parameterize the manifold by embedding it into a low dimensional Euclidean space such that the coordinate along one axis is given by the known parameter and the coordinates along the remaining are statistically independent of the known parameter. The need for such an algorithm arises naturally in a number of medical imaging applications. A common situation is that extra meta-data is received during image acquisition, such as the pose of the camera or information about the state of the object, including breath or heartbeat phase. We give several different approaches for addressing this problem based on manipulation of coordinates before applying Manifold Learning, manipulation of a distance or kernel matrix used directly within a Manifold Learning algorithm, or manipulation of coordinates after applying Manifold Learning. Each approach offers clean formalisms which allow extra information to improve the manifold learning result.*

## 4.1 Introduction

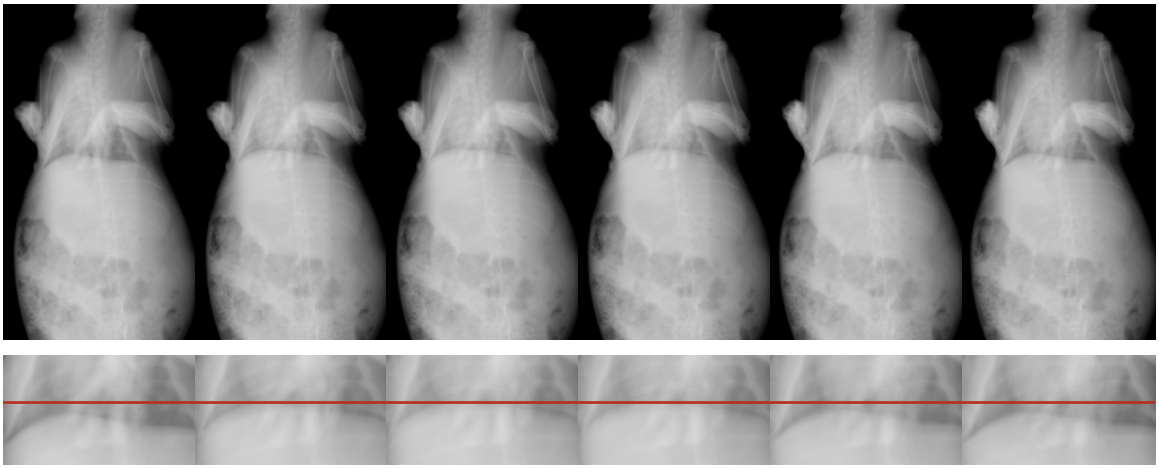
Motivated by our work with medical imaging datasets, where extra information such as belt measurements or heart monitors are available, in addition to medical images, we look at a more general problem of creating an embedding when part of the desired output parameterization is known a-priori. Often, this known parameter correlates with large image changes. For example, in 4D CT data, an external belt measures the patient breathing and breathing is also the dominant cause of image change. Our problem is to use knowledge of the parameter to better account for remaining variations, or equivalently to create an embedding in which the known parameter is shown on one axis and the remaining variation, which is not accounted for by the known parameter, is represented on the other axes. We call this problem Partially Unsupervised Manifold Learning (PUML).

Consider the Conebeam CT images of a live rabbit in figure 4.1. A set of 578 images are taken from equally spaced, known angles comprising a full circle around a rabbit. The primary cause of variation in the image set is due to changes in viewing angle. During image acquisition, the rabbit is breathing and completes a breath cycle once every several frames. Our goal is to use the known viewing angle parameter to ignore the viewing angle variation in the images and to better parameterize the much smaller changes due to breathing. We formalize the following problem: given a set of $N$ data samples consisting of an input column vector and a known parameter value, $(X_{\bullet 1}, a_1), (X_{\bullet 2}, a_2), \ldots, (X_{\bullet N}, a_N)$, solve for $Z$, a low-dimensional parameterization which best preserves the structure of the high dimensional data, while simultaneously requiring that the output coordinates be independent of $\vec{a}$.

Section 4.2 presents related work within manifold learning. Section 4.3 presents the datasets on which we will evaluate our algorithms. Section 4.4 introduces the notation we use and gives a detailed problem formulation. Section 4.5 gives a description of the Isomap algorithm: this algorithm makes use of all the data structures which we modify to produce PUML algorithms. Section 4.6 presents various approaches to solving PUML. Section 4.7 shows results of our algorithms on various datasets.



Long Term Variation Due to Known Camera Angle (578 Frames)



Short Term Variation Due to Breathing (6 Frames)

Figure 4.1: Top: Conebeam CT images of a rabbit. The main cause of variation in the images is due to the viewing angle change as the camera rotates completely around the rabbit. Bottom: Images of the rabbit during one breath cycle comprising 6 frames. Although the viewpoint changes a little, the dominant changes in these images is due to breathing. In the close up images shown below the main images, the diaphragm of the rabbit can be seen to rise slightly. A horizontal red line is drawn across the images for comparison.

## 4.2 Related Work

Many methods have been developed to work with non-linear manifolds and to do non-linear dimensionality reduction; a survey of methods is given in [88]. We present a summary of the major methods available in dimensionality reduction and related algorithms. We classify all current methods into the following categories: (1) Spring Networks, (2) Self Organizing Maps, (3) Probabilistic Mappings, (4) Neighborhood Graph Based Methods, (5) Tangent Space Alignment Methods, (6) Tangent Space Embedding Methods, (7) Cyclic Manifold Embedding Methods, and (8) Multiple Data Source Embedding Methods. We will build off the Isomap algorithm when presenting our PUML algorithms, since Isomap uses each of the data structures which we are able to modify to incorporate a known parameter into. We will also use Canonical Correlation Analysis (CCA) directly within our local orthogonalization method.

### 4.2.1 Spring Networks

The first methods for dimensionality reduction were based on preserving pairwise distances between points while linearly embedding them into a lower dimensional space. The linear transform which optimally preserves distances between points is found by Metric Multidimensional Scaling (MDS) and equivalently Principal Component Analysis (PCA). Since these methods can only model linear manifolds, many extensions have been proposed to handle non-linear manifolds. These include extending MDS to use only ordering data and to minimize different stress functions. In general, these methods can be thought of as solving for the rest state of a network of springs which represent the distances between points. Such methods are still under development; recent work such as DrLIM uses special spring like constructs which either repel points only when they are close together or always attract points toward each other [39].

### 4.2.2 Self Organizing Maps

Self Organizing Maps (SOM) were an early attempt to model non-linear relationships between data points [47]. This method lays down a grid of points in the embedding space with each grid point modeling a portion of the data manifold. A smooth mapping between the manifold and the grid points in the embedding space is then trained. Generative Topographic Mapping (GTM) extends this method into a probabilistic framework [12]. The grid points in GTM each use a Gaussian distribution to jointly model the probability that a point in the high dimensional space is on the manifold.

### 4.2.3 Probabilistic Mappings

Several methods modify standard PCA using the kernel trick. The kernel trick in its basic form is the observation that a variety of kernel functions, including the Gaussian function, can be considered as the inner product between points in a high (possibly infinite) dimensional space. Since an inner product between points is the only thing necessary for several algorithms, such as MDS, to function, this has the advantage of working in a high dimensional space where the data may be more easily separable, without actually having to compute the location of points in the high dimensional space. Kernel PCA creates an embedding using PCA except that similarities between points are computed from a kernel function rather than the standard Euclidean inner product [77].

The kernel trick is used in the reverse direction in Gaussian Process Latent Variable Models (GP-LVM) [49, 50]. As in GTM the embedding space directly models the data manifold. However, instead of a grid of points, each data point has a corresponding latent variable which models a portion of the manifold based on its position in the embedding space. The position of the latent variables is computed via an optimization problem which maximizes the likelihood of the data points given the way in which latent variables model the manifold. Since GP-LVM only finds local minima, choosing a good initial position for the latent variables is important. This work has been extended to simultaneously solve for the intrinsic dimensionality of the manifold and an embedding [34].

### 4.2.4 Neighborhood Graph Based Methods

Most of the remaining methods make use of a neighborhood graph in which each point is connected to its surrounding neighbors. Usually this is done by connecting each point to its $k$ nearest neighbors; however, this can also be done in other ways, such as by connecting all points which are separated by less than $\epsilon$ distance. The weight for each edge varies by method but is generally related to the Euclidean distance between the two points. There is a tendency for neighborhood graphs in high dimensional data to have hub structures in which some data points have many neighbors, while most have few [71].

One of the most well known manifold learning algorithms is Isomap [84]. We present this algorithm in more detail in section 4.5. Isomap uses the neighborhood graph to estimate geodesic distances between points. It assumes that the manifold sampling is dense enough that each local neighborhood in the graph can be modeled linearly. The geodesic distances over the manifold are approximated by computing shortest paths through the neighborhood graph of Euclidean distances. The pairwise geodesic distance matrix is then used to create point positions in a low dimensional Euclidean space using metric MDS. This method preserves relationships between far away points explicitly. Isomap is only guaranteed to recover an accurate parameterization when the dataset is isometric to Euclidean space and sufficiently sampled [25, 26].

Locally Linear Embedding (LLE) [73] uses local properties to construct an embedding using spectral techniques. First, a weight matrix which optimally reconstructs the position of each point from the location of its neighbors is found. Second, an eigenvalue problem is solved to find the embedded points which best preserve the relationship between points encoded in the weight matrix. This method does not explicitly preserve any relationship between far away points.

In Laplacian Eigenmaps, the neighborhood graph is computed and the smallest eigenvectors of the graph Laplacian used to create an embedding space [9, 92]. This method produces an embedding where distances between points are related to the expected number of steps in a random walk between the points in the neighborhood graph when the distance of the edges is considered as a transition probability. This method

is the basis for spectral clustering in which k-means is generally run in the embedding space to create a clustering.

Hessian Eigenmaps (HLLE), also called Hessian Locally Linear Embedding, is a method analogous to the Laplacian Eigenmaps method but using an estimated Hessian matrix at each data point in place of the graph Laplacian [24]. This method is particularly well suited to punctured manifolds in which large holes are present.

Maximum Variance Unfolding (MVU) preserves local nearest neighbor distances while maximizing the total variance of the data points [96, 97]. In effect this takes the ends of the manifold and stretches them as far apart as possible without tearing the neighborhood graph of the manifold. This problem is formulated as a semidefinite program and can be optimized efficiently. Slack variables can be added to allow some of the local distances to be violated.

### 4.2.5 Tangent Space Alignment Methods

This class of manifold learning algorithms focuses on aligning local neighborhoods of the manifold. The first major method of this type, Charting [13], uses a local manifold dimensionality estimation technique to determine the intrinsic dimension of the manifold based on its data sampling. A linear representation of each local neighborhood is created using a maximum a posteriori (MAP) estimate of a Gaussian mixture model. This method uses more global data in the construction of its local manifold estimates than simply taking the local PCA space at each location. A coherent embedding of the whole manifold is created from the local representations by projecting them linearly into an embedding space such that points which are in more than one chart are mapped to the same location. Similar methods include Global Coordination [74] and Local Tangent Space Alignment (LTSA) [105]

### 4.2.6 Tangent Space Embedding Methods

Another approach is based on picking a central point from which to construct an embedding space. Logmap uses PCA over the neighborhood of the central point

to create a tangent space of the manifold at that point [15]. All other data points are then projected into this tangent space. This is done by determining the polar coordinates of the point with respect to the center point. The radius from the center is estimated, as in Isomap, by the geodesic distance in the neighborhood graph. The direction from the center point is determined by finding the gradient of the distance in the neighborhood of the center point. This gradient is computed by finding the geodesic distances to the center point's neighbors and performing a regression to fit a plane to these values.

This method is able to tear manifolds which are not homeomorphic to an open disk and embed them in a plane. Depending on the dataset, this method of dealing with cyclic structures may be preferable to distorting the entire manifold. Cyclic structures are simply broken apart at the part of the cycle furthest from the center point. To help create a clean cut at edge points, Logmap uses Random Sample Consensus (RANSAC) to perform a robust regression and estimate the direction to the center. A drawback of Logmap embeddings is their reliance on a single center point. With many manifolds the quality of the embedding decreases quickly with distance from the center point, making it difficult to produce a single good embedding of the manifold using this method. A modification of Logmap constructs a simplicial mesh representation of the manifold [33] and then positions points far from the central point by preserving angles within the simplicial mesh [55].

A similar method to Logmap, ExpMap, can be used to apply textures to 3D objects [76]. This method reasons explicitly about the tangent space at sample points to convert coordinates from one tangent space to the next. Unfortunately, this approach is only possible in a three dimensional space, where a 2D tangent plane can be rotated onto another 2D tangent plane in only one way.

### 4.2.7 Cyclic Manifold Embedding Methods

Cyclic manifolds present a problem for many manifold learning algorithms. One method to address this problem is to explicitly determine a set of edges at which to cut the graph so as to remove cycles. A robust method for creating such a cut removes all chordless cycles larger than a user specified threshold [51]. Chordless cycles are

cycles which do not have any edges between nodes except those between neighbors: they are the discrete equivalent of non-contractible loops. This method has been used to embed distorted cylinders [23]. In this case a graph is cut, then cloned, and cut again. The cloning is accomplished by creating two identical cut graphs and replacing the cut edges with edges which go from one copy to the other. The resulting graph, once cut again, is embedded using Isomap.

Other methods are geared directly towards parameterizing cyclic manifolds. Some work has been done on parameterizing manifolds of any genus by coordinates along handle and tunnel loops [22], parameterizing manifolds with non-Cartesian latent space in back constrained GP-LVM models [86], and parameterizing spherical manifolds [70]. Our PUML algorithms are able to compute over a manifold in which the known parameter is cyclic, even when the host manifold learning algorithm (such as Isomap) does not support cyclic parameters.

### 4.2.8 Multiple Data Source Embedding Methods

There are several Manifold Learning algorithms which consider more than one data source. Canonical Correlation Analysis (CCA) addresses a multiple manifold embedding problem in a linear setting [6]. The approach of CCA, using techniques similar to MDS and PCA, finds two matrices which linearly transform the input spaces so that they are most correlated in a small number of dimensions. The result is a low dimensional embedding space into which there is a mapping function from each input manifold. CCA has been extended to a non-linear formulation which seeks a consistent, non-linear transformation of the input manifolds into a common space. Approaches have been based on the GP-LVM framework [29, 52] and on Laplacian Eigenmaps [91]. We solve a similar problem where instead of trying to find an embedding of the manifold which is most correlated with the known parameter, we try to find an embedding of the primary manifold which is independent of the secondary manifold (which we call the known parameter) but still preserves structure within the primary manifold.

Another example of a related problem is colored Maximum Variance Unfolding, which takes as input both points from a manifold and a class label for each data point [81].

Extra constraints are added to the normal MVU formulation to find the embedding which best preserves the ability to distinguish between different label classes. This has the effect of reducing the dimensionality of the data space while still preserving the ability to cluster the data. The problem of creating specially tuned embeddings based on label information has also been studied in the context of kernel learning [56].

In our time sequence analysis, we remove long term trends to allow the extraction of repetitive short time scale variations. The reverse problem of extracting the general trend has been studied to determine the effect age has on brain shape in MRI scans [21].

## 4.3 Datasets

The PUML problem is motivated by problems in applying manifold learning to natural image data sets. This section introduces the 3 datasets shown in figure 4.2. Each dataset has a known parameter which is acquired in tandem with the dataset. The dataset varies due to both this known parameter and secondary variation. For each dataset a ground truth value for the secondary variation is also acquired.

The first example is from a Conebeam CT scanner in which images of a live rabbit are captured from 578 equally spaced angles comprising a full rotation around the rabbit. The rabbit is breathing during image acquisition, completing a full breath cycle once every few frames. This dataset has a cyclic dimension because the CT scanner completes an entire rotation around the rabbit. We show results on both the full dataset and a set of 144 images comprising a quarter of a circle around the rabbit. The known parameter is the rotation angle of the CT scanner from which the image was acquired (rescaled to the range [-1,1]). The ground truth for the secondary variation due to breathing was determined by manually marking the position of the diaphragm in each image.

The second dataset is a sequence of images from a short axis echocardiogram of the left ventricle of the heart. This data set varies due to "camera drift" (as the ultrasound probe is somewhat unsteady), and due to the heart phase. Ground truth for the heart phase was determined by manually measuring the width of the ventricle from left to

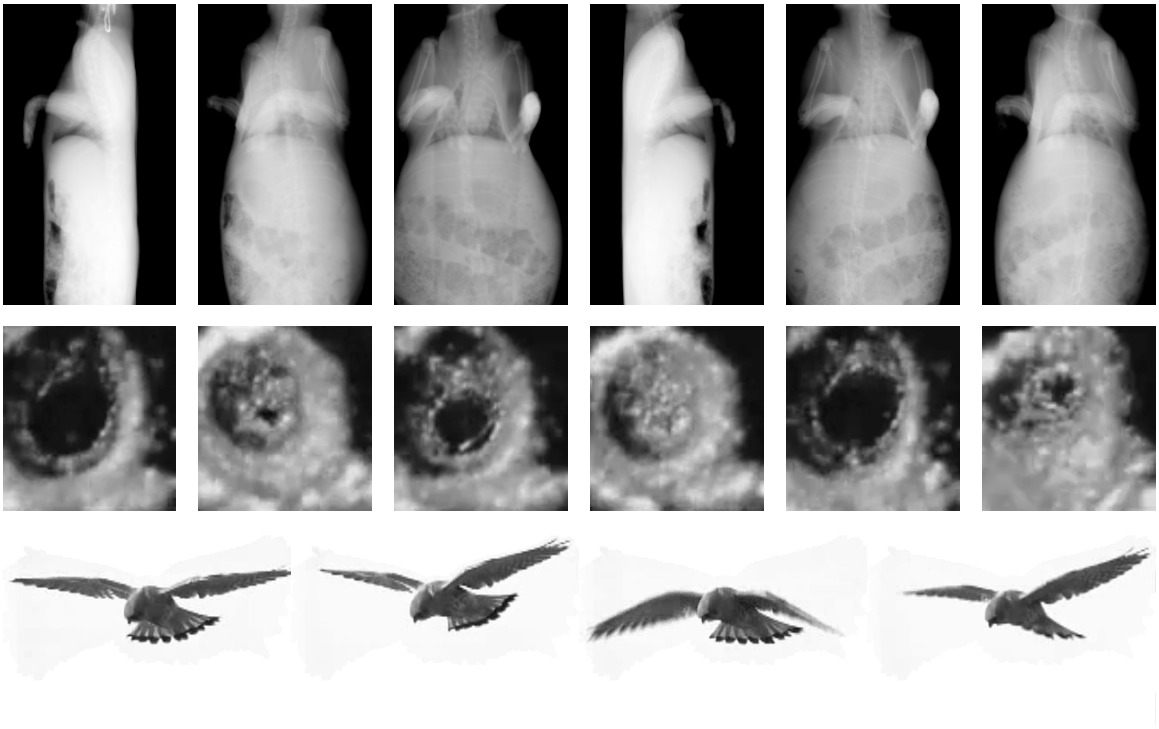


Figure 4.2: First row: A Conebeam CT of a breathing rabbit. This is a 2D manifold with projection angle and lung volume as dimensions. Second row: The left ventricle of the heart as seen in a short axis echocardiogram. Third row: A kestrel in flight with its head registered to approximately the same position and size in each frame.

right and from top to bottom and adding these two numbers. The beating is very fast compared to the time scale of camera drift; thus, this dataset was included to explore cases where, over long time intervals, the time of image capture serves as a good proxy for the distracting variable.

The last dataset is a video sequence of a kestrel in flight. This was registered so that its beak maintains a relatively fixed position. This dataset is included to explore the situation when there are many remaining degrees of freedom, including the position of both wings, the tail, and the pitch of the body. The known parameter is the timestamp of the video frame, which again serves as a proxy for the distracting variable. Ground truth is determined by measuring the vertical position of the bird's right wing.

## 4.4 PUML Problem Formulation

The problem formulation for PUML is as follows. Given a set of $N$ data samples consisting of an input column vector and a known parameter value, $(X_{\bullet 1}, a_1), (X_{\bullet 2}, a_2), \ldots, (X_{\bullet N}, a_N)$, solve for $Z$, a low-dimensional parameterization which best preserves the structure of the high dimensional data. We use a dot notation to mean all entries in a column or row of the matrix: $Z_{\bullet j}$ are all the low dimensional embedding coordinates of a particular sample $j$, while $Z_{i\bullet}$ is the row vector consisting of the $i$th low dimensional embedding coordinates for all the samples. In our problem formulation we further require that the coordinate of the samples in each output dimension $Z_{i\bullet}$ be independent of $\vec{a}$.

$$\underset{Z}{\operatorname{argmin}} \sum_{(i,j)\in E} \left\| d_{ij} - \|Z_{\bullet i} - Z_{\bullet j}\| \right\|^2 \textbf{ and } \forall k,\, Z_{k\bullet} \textbf{ independent of } \vec{a} \tag{4.1}$$

In this equation, $d_{ij}$ is the distance between samples $i$ and $j$ in the input data. Although this might be Euclidean distance, it can also be measured in different ways, such as an estimate of geodesic distance over the input manifold. The set $E$ over which the sum is computed consists of all samples which will be compared: this set might be every pair, or it might only include neighboring pairs of samples. For the Isomap algorithm, $E$ is the set of every pair of samples and $d_{ij}$ is computed as the shortest path distance between samples in the neighborhood graph. The last part is that

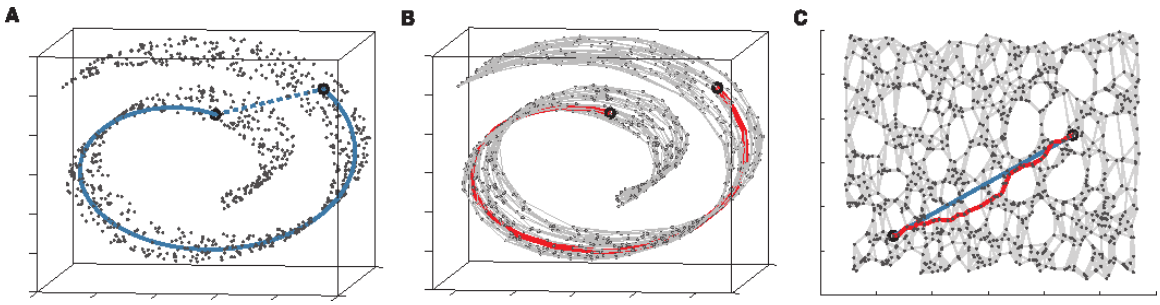



Figure 4.3: (A) The Euclidean distance (dotted blue line) between two points is not constrained to lie within the manifold and hence does not reflect the true structure of the data. The geodesic distance (solid blue line) is the shortest path between two points within the manifold. (B) In Isomap, the geodesic distance is estimated as the shortest path distance between two points in the neighborhood graph. (C) The Isomap embedding of the data in two dimensions is shown with the shortest path distance and geodesic shown. This figure was taken from [84].

the output coordinates should be statistically independent of the known coordinates. This slightly abuses notation by considering both $Z_{i\bullet}$ and $\vec{a}$ as single random variables instead of as vectors of samples chosen from those random variable. In our approach we will use different methods to create coordinates which are independent of $\vec{a}$. For example, the global orthogonalization method replaces the independence criteria with decorrelation, which is explicitly enforced.

## 4.5 Isomap Algorithm

The algorithms we present are modifications which must be added to other Manifold Learning algorithms to produce a complete algorithm. We use as an example the Isomap algorithm, for its effectiveness, simplicity, and use of all the data pieces which we are able to modify to create a PUML algorithm.

The essence of the Isomap algorithm is the use of metric Multidimensional Scaling (MDS) on a modified distance matrix which estimates the geodesic distance between points in the input manifold instead of direct Euclidean distance.

The first part of Isomap is to estimate the geodesic distance between points. A geodesic is the shortest curve between two points which remains on a manifold. Since

the manifold is curved in high dimensional space, the geodesic will generally be a curve. Figure 4.3 shows the difference between direct Euclidean distance in the high dimensional space and geodesic distance between two points. Isomap begins by finding the nearest neighbors of every point in the dataset (we will assume $k$ nearest neighbors are used). A neighborhood graph is created in which neighbors are connected with edges of length equal to the Euclidean distance between them. The intuition is that locally on a manifold the space is flat and Euclidean distance estimates the geodesic distance well. However, between points that are far away, the geodesic distance between points is very different from the Euclidean distance. An estimate of the geodesic distance is computed as the shortest path distance between two points in the neighborhood graph. The matrix $D$ is constructed where the entry $d_{ij}$ is the shortest path distance between sample $i$ and $j$.

This distance matrix is then converted into a kernel matrix (also called a similarity matrix) $K$ by multiplications with the centering matrix $H$.

$$K = HD^2H \tag{4.2}$$

$$H_{i,j} = \begin{cases} 1 - \dfrac{1}{n}, & i = j \\ -\dfrac{1}{n}, & i \neq j \end{cases} \tag{4.3}$$

$n$ is the number of samples. If $D$ is a true distance matrix in which distances are non-negative and the triangle inequality holds, then the matrix $K$ is a positive semi-definite matrix which can be decomposed as $K = Y^TY$. The output coordinates $Y$ can be solved for using an eigenvector decomposition of the matrix $K$. The columns $Y_{\bullet i}$ of $Y$ are generally arranged in order of decreasing corresponding eigenvalue which means they have the property that the first $m$ columns of $Y$ have the optimal reconstruction of $K$ of any rank $m$ matrix. These are the output coordinates of Isomap.

## 4.6 PUML Methods

Each of the presented methods modifies the data at a particular stage of the Isomap algorithm. Figure 4.4 shows a diagram of the stages of the Isomap algorithm on the

left and at which stage each of the PUML methods which we will present modify the data.

Section 4.6.1 presents an algorithm which modifies the computed distance measures by observing that a portion of the distances should depend on the known parameter, and a portion on unknown factors. Section 4.6.2 modifies this approach to be used within the local neighborhoods of points. Section 4.6.3 presents an algorithm which post-processes a parameterization to find variation in the dataset which are independent of the known parameter. Section 4.6.4 changes the distance measure in local neighborhoods to make them independent to changes in the known parameter. Section 4.6.5 performs pre-processing to remove the long term trends in the dataset before obtaining a parameterization using a standard manifold learning method.

Of the methods which will be presented, the trend subtracted PUML method (presented in section 4.6.5) is by far the most effective; however, the other methods are provided for context and motivation. Each method is evaluated in section 4.7, which also provides a visualization of which pixels in the images are most responsible for variation in the dataset.

### 4.6.1 Kernel Modification by Distance Subtraction

Our first approach to PUML modifies the distance matrix to create a new distance matrix which is independent of the known parameter. We start with the intuition that the distance between points in the embedding space can be explained using a right triangle, as displayed in figure 4.5. One of the legs of the triangle is the distance between points in the known parameter, the other is the distance in the output embedding, and the hypotenuse is the observed distance between samples $d_{ij}$.

We assume that the distance between samples is composed of two orthogonal parts, one dependent on the known parameter coordinates of the samples $a_i$, the other on the remaining unknown output coordinates $Z_{\bullet i}$ as shown in figure 4.5. The distance matrix of the known parameter $D_a$ consists of pairwise distances between samples in known coordinates. Similarly, we write the observed distance matrix as $D$ and the distance matrix between points in output embedding space $Z$ as $D_z$. The Pythagorean

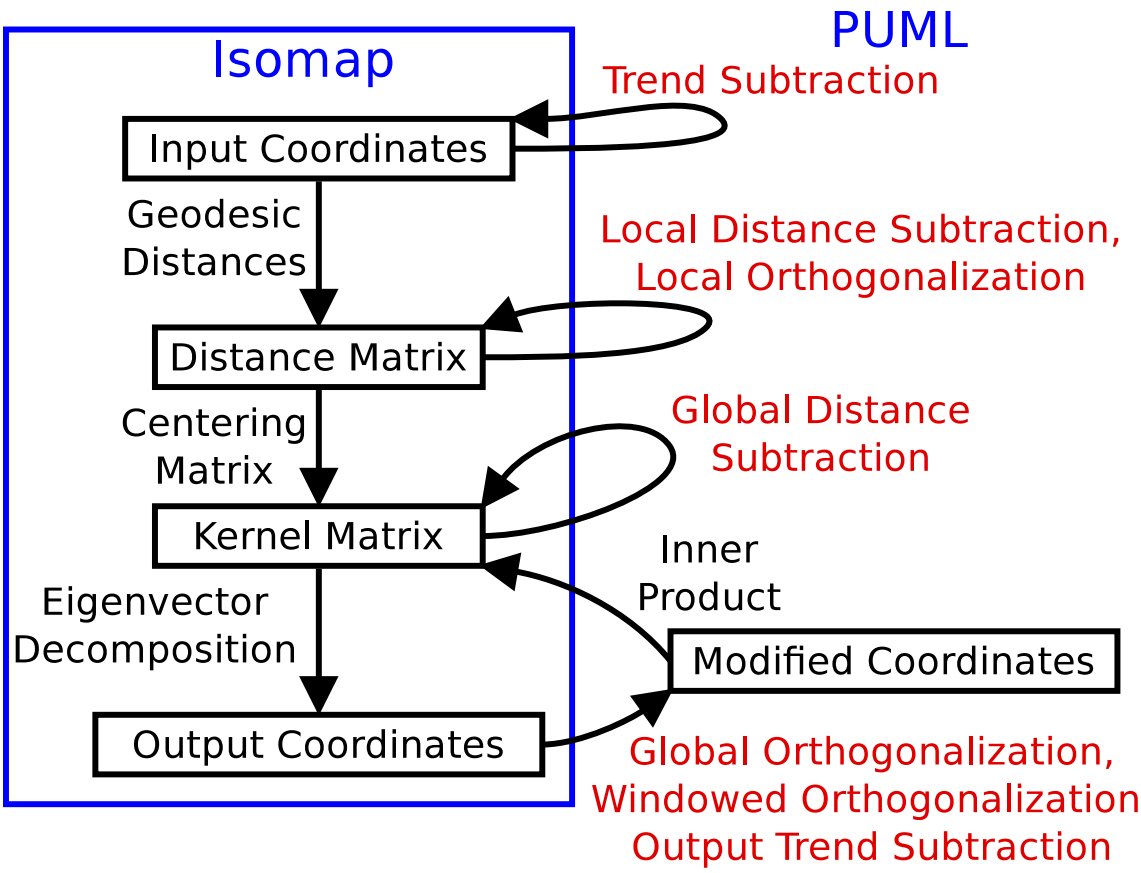



Figure 4.4: Flowchart depicting the different PUML methods presented in this chapter (in red). Boxes represent data types, while arrows are algorithms. Isomap starts from the input coordinates, computes a distance matrix by estimating the geodesic distances by shortest path in the neighborhood graph and then converts this to a kernel matrix which is used to produce output coordinates through an eigenvector decomposition. Each of the presented methods adds a piece to one of the steps in this algorithm.

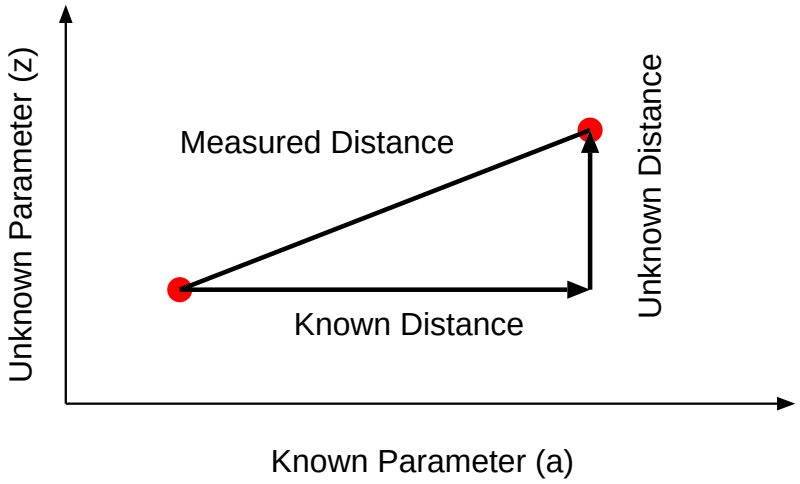



Figure 4.5: The distance between two points can be decomposed as a known distance based on the points known coordinates, and an unknown distance orthogonal to the known distance.

theorem requires that the following equation hold.

$$D_z^2 = D^2 - \alpha D_a^2 \tag{4.4}$$

The square of the unknown distance is the square of the measured distance between points minus the square of the known distance between points. We assume that we know $D_a$ up to a scaling factor $\alpha$. The scaling factor is necessary because the known parameter is in a different coordinate system than the data is in.

We transform the distance matrix $D_b$ into a kernel matrix $K_b$ as outlined in the last section. If the distances come from a Euclidean space, then $K_b$ will be a positive semi-definite matrix. Unfortunately, for large unknown scaling factor $\alpha$, the resulting matrix $K_b$ may not be positive semi-definite.

$$K_b = K - \alpha K_a \tag{4.5}$$

Finding the correct $\alpha$ can be done by analyzing the eigenvalues of the matrix $K_b$. We wish to remove the degrees of freedom corresponding to an entire dimension in the input coordinate system, since it is assumed that the known parameter accounts for this freedom. This means that the rank of the kernel matrix must be reduced by one and consequently that we need to set $\alpha$ so that an eigenvalue becomes zero. In

practice, it is easier to minimize the sum of the squared eigenvalues of $K_b$ which is equivalent to the trace of $K_b^2$.

$$\arg\min_{\alpha} tr(K_b^2) \quad s.t. \quad K_b = K - \alpha K_a \tag{4.6}$$

Alternatively, a constant term can be added to a non-semi-definite matrix to create a semi-definite matrix as is done in kernel Isomap [19]. Notice that although we have motivated this algorithm by considering distances, the algorithm itself can be performed directly on the kernel matrix. We call this method global subtraction, in contrast to the local subtraction method in the next section.

Figure 4.6(c) shows the result of applying this method to a bent helicoid-like manifold. The known parameter is the distance from the centerline of the helicoid, and the secondary variation, shown in color, is the distance along the centerline. Global subtraction is able to produce a better parameterization than Isomap given the known parameter. A more detailed analysis on multiple datasets is given in section 4.7.

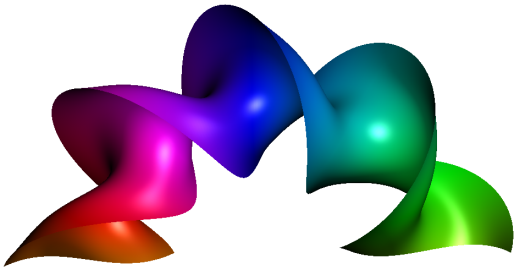
### 4.6.2 Applying Distance Subtraction Locally

In the last section we modified the distance matrix as a whole, after the shortest path computation in Isomap (see section 4.5). This lead to an algorithm which works directly on the kernel matrix. However, we can do distance subtraction before the shortest path computation as well. The motivation for doing the distance subtraction at this stage is that it is more likely that the triangle considered in figure 4.5 is correct at a local scale, where each of the edges are a direct distance between neighbors, rather than a shortest path computation through many hops in the neighborhood graph.

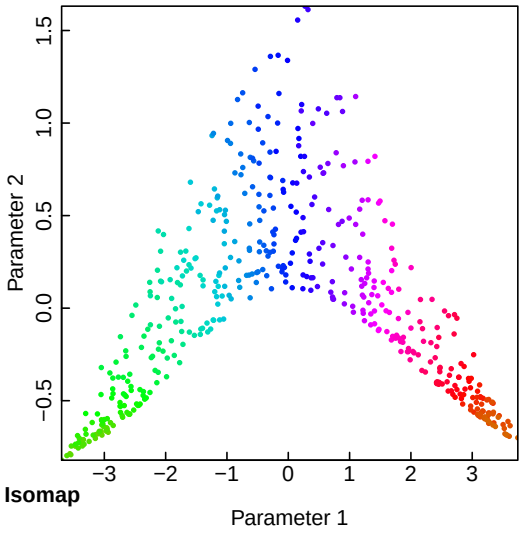
As before we subtract distances; however, we now think of $D$, $D_a$, and $D_z$ as all pairs Euclidean distance, rather than geodesic distances.

$$D_z^2 = D^2 - \alpha D_a^2 \tag{4.7}$$

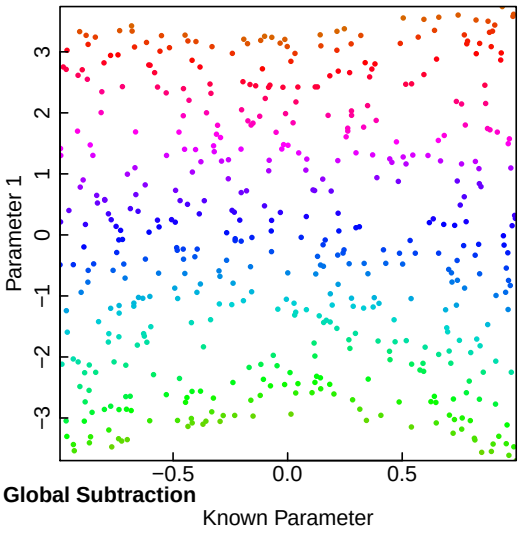
Geodesic distances are then estimated by a shortest path distance through the neighborhood graph, using the modified distances of $D_z$. This matrix is then converted into a kernel matrix, and coordinates extracted by an eigenvector decomposition in



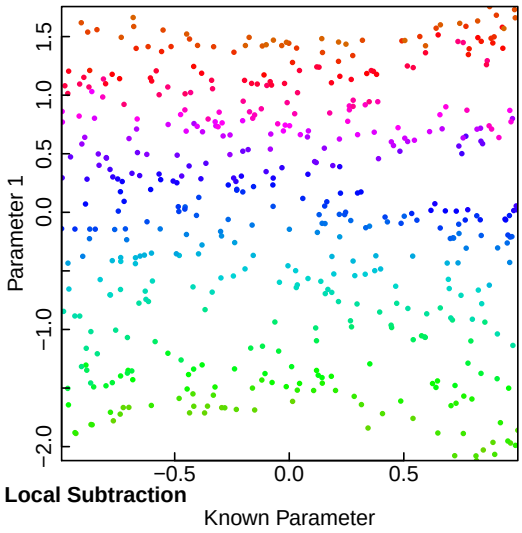
(a) Bent Helicoid-like Manifold




(b) Isomap Embedding

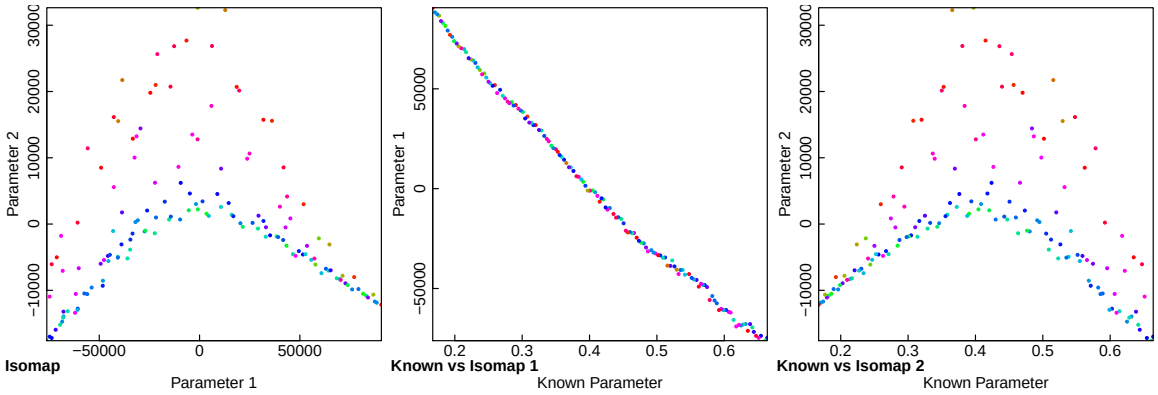



(c) Global Subtraction PUML Method




(d) Local Subtraction PUML Method

Figure 4.6: Local and global subtraction methods applied to a bent helicoid-like manifold. The known parameter is the distance from the center line of the helicoid, while the secondary variation, shown in color, is the distance along the center line. (b) For comparison the Isomap embedding of the manifold is shown. (c) Global subtraction as described in section 4.6.1. (d) Local subtraction as described in section 4.6.2.




(a) Isomap (b) Isomap Coordinate 1 (c) Isomap Coordinate 2

Figure 4.7: (a) X-ray images of a rabbit cropped around the lung taken from various angles (spanning $\frac{\pi}{2}$ radians) are embedded in a two dimensional space using Isomap. (b) The first and (c) second Isomap parameter is plotted against the viewing angle. The color in all plots is ground truth diaphragm position.

the normal way. We term this algorithm local distance subtraction, in contrast to the global distance subtraction method of the previous section.

In figure 4.6, the results of applying local subtraction to the synthetic bent helicoid-like manifold are shown. This method is also able to parameterize the remaining variation, given the known parameter. A more detailed analysis on multiple datasets is given in section 4.7.

### 4.6.3 Orthogonalized Manifold Learning

We now turn our attention to methods which post-process the output coordinates produced by a manifold learning algorithm to produce coordinates independent of the known parameter. Without special action a manifold learning algorithm will parameterize all relevant sources of variation, including both variations which are dependent on the known parameter and those that are not. Figure 4.7(a) shows the Isomap embedding of the rabbit over $\frac{\pi}{2}$ radians dataset. We will denote the Isomap coordinates by $Y$. Also presented are two plots of the first and the second Isomap parameter against the known parameter. Notice that $Y_{1\bullet}$ the first Isomap parameter correlates almost perfectly with the known parameter $\vec{a}$. Isomap is providing information which we already have available. We are much more interested in the

second Isomap coordinate, which parameterizes variation in the dataset which is not captured by the known parameter.

In the problem statement shown in equation 4.1, we wish to find output coordinates which are independent of the known parameter. In the orthogonalization approach we replace this criteria with decorrelation, a necessary condition for independence. The desired output coordinate, $Z_{i\bullet}$, should not be correlated with the known parameter $\vec{a}$. This is accomplished by ensuring that the vectors $Z_{i\bullet}$ are orthogonal with $\vec{a}$.

Given a set of output vectors $Y_{i\bullet}$ of coordinates as produced by Isomap, we can generate orthogonal vectors by orthogonalizing the following matrix.

$$A = \begin{pmatrix} \vec{a} & Y^T \end{pmatrix} \tag{4.8}$$

The Gram-Schmidt process can be used to create a new matrix $B$ in which each column is orthogonal to every other column and to $\vec{a}$.

$$\begin{pmatrix} \vec{a} & B^T \end{pmatrix} \tag{4.9}$$

The Gram-Schmidt process is an iterative algorithm where the first step is to project each column onto the first column and subtract that projection from it, thus orthogonalizing each column with respect to the first column. The next iteration orthogonalizes each column right of the second with the second. This process continues until all columns are orthogonal to each other. It is common to also normalize each column during this algorithm; however, this should not be done in our application since the relative importance of each column should be maintained throughout the process. This produces a new matrix where every column $B_{i\bullet}$ is orthogonal to every other $B_{j\bullet}$ and also orthogonal to $\vec{a}$.

After the orthogonalization process, the first component $B_{1\bullet}$ might have very little variation left. This is the case for datasets like in figure 4.7, where the first Isomap coordinate $Y_{1\bullet}$ is almost perfectly correlated with $\vec{a}$. We use Principal Component Analysis (PCA) to compute the components in $B$ which have the most variance remaining. This is equivalent to computing a new kernel matrix as the inner product of $B$: $K = B^T B$. The eigen decomposition of $K$ yields the final output coordinates

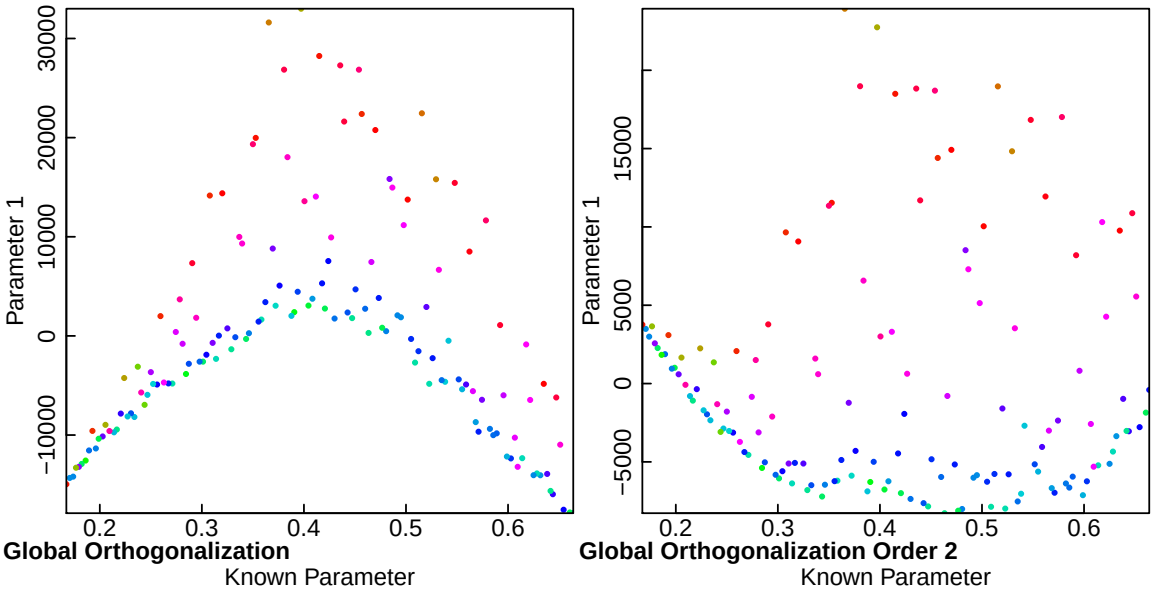



Figure 4.8: The result of orthogonalization on the same dataset as figure 4.7. Left: only a first order component is removed through orthogonalization. Notice the similarity between this output and the original second Isomap parameter in figure 4.7(bottom right). Right: a first and second order component is removed.

$Z$ which are now not correlated with the known parameter $\vec{a}$. Figure 4.8 shows the output for the same manifold as figure 4.7.

#### 4.6.3.1 Higher Order Orthogonalization

We motivated the last section by observing that decorrelation is a necessary condition for independence and ensuring that the output parameters are not correlated with the known parameters. However, the condition of independence implies that there are also no higher order correlations present. In this section, we extend the concept of orthogonalization to remove higher order correlations. We use the same Gram-Schmidt orthogonalization process, but add higher order terms.

$$A = \begin{pmatrix} \vec{a} & \widehat{\vec{a}^2} & \cdots & \widehat{\vec{a}^n} & Y^T \end{pmatrix} \tag{4.10}$$

Where $\widehat{\vec{a}} = \vec{a} - \overline{\vec{a}}$ denotes a vector minus its mean (and $\vec{a}$ already has mean zero). After orthogonalization we obtain a matrix $B$ in which each column is orthogonal and

decorrelated with $\vec{a}$ to the $n$th degree. As with the previous method, we use this matrix to create a new kernel matrix, and an eigen decomposition to produce the output coordinates $Z$. Figure 4.8(right) shows results for second order orthogonalization on the rabbit over $\frac{\pi}{2}$ radians data.

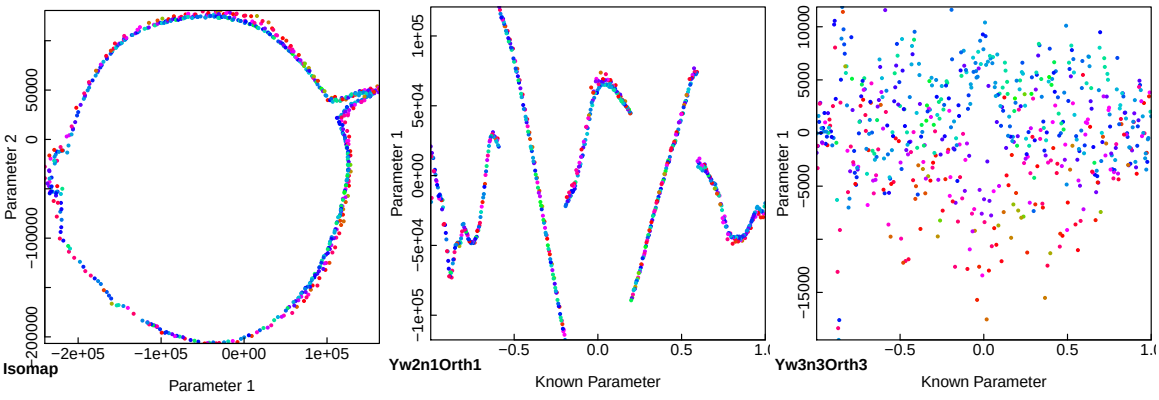
#### 4.6.3.2 Windowed Orthogonalization

In this section, we take a different view of the condition of independence. We note that a necessary condition for independence is to not be correlated over any set of samples. We break down the dataset into windows of samples. In each window, the output coordinates of those samples should not be correlated with the known parameters. Gram-Schmidt orthogonalization performed on the following matrix will ensure that this is true.

$$\begin{pmatrix} \widehat{\vec{a}(1..\frac{N}{2})} & \vec{0} & \top & \top \\ \vert & \perp & \vec{0} & \vert \\ \vert & \top & \perp & Y^T \\ \vec{0} & \widehat{\vec{a}(\frac{N}{4}..\frac{3N}{4})} & \vert & \vert \\ \perp & \perp & \vert & \perp \\ & \vec{0} & \widehat{\vec{a}(\frac{N}{2}..N)} & \end{pmatrix} \tag{4.11}$$

By changing the window sizes and window overlap, trends over the known parameter occurring at different scales are found and removed, leaving a signal independent of the known parameter. Higher order terms can be used in the same way as earlier. In figure 4.9 we can see that this method can be effective, but also creates artifacts at the edges of windows.

A danger when using this method is that the data will be orthogonalized with respect to too many different vectors. Each orthogonalization vector spans a portion of the data space, given enough such linearly independent vectors, any data vector can be represented. When the dataset is orthogonalized with respect to too many vectors, all variation within the dataset is removed, including the variation of interest. This can be particularly problematic when using higher order terms, since even only using second order terms for each window doubles the number of vectors being used in the orthogonalization.

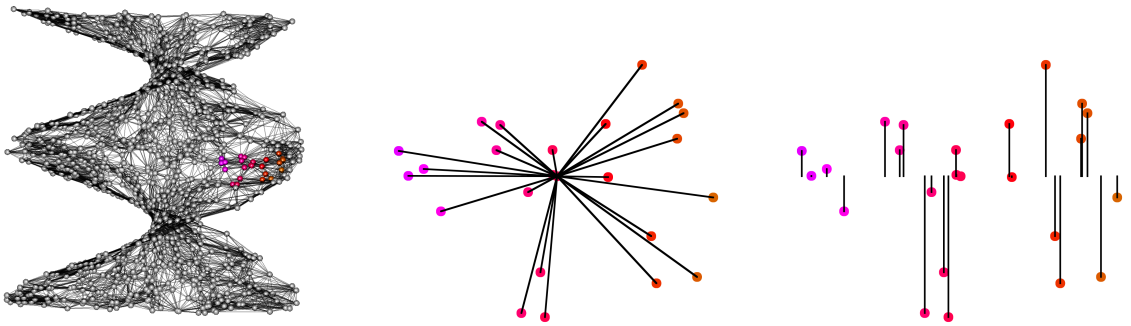



(a) Isomap

(b) 6 windows, Overlap of 2, First order decorr

(c) 12 windows, Overlap of 3, Third order decorr

Figure 4.9: The rabbit dataset over all angle views. (a) Isomap parameterization. (b) Windowed first order orthogonalization over large windows causes discontinuity artifacts along the ends of the windows (the windows overlap such that every sample is within 2 windows). (c) Under the right conditions windowed orthogonalization removes the dependence of the output coordinates on the known parameter. In this example, third order orthogonalization is used over 12 windows in which each sample is within 3 windows at once. In this case, windowed orthogonalization is able to recover a breath signal, which is visible because the y-axis now captures some of the color variation (which encodes ground truth breath phase).



(a) Pick Neighborhood (b) Find Correlation Direction (c) Remove Projection

Figure 4.10: The steps of the local orthogonalization algorithm. (a) A neighborhood graph is computed and for each local neighborhood the following steps performed. (b) The local neighborhood is oriented such that the direction most correlated with the known parameter is from left to right. (c) the points are projected into the subspace not spanned by the direction most correlated with the known parameter. These new distances should be independent of the known parameter.

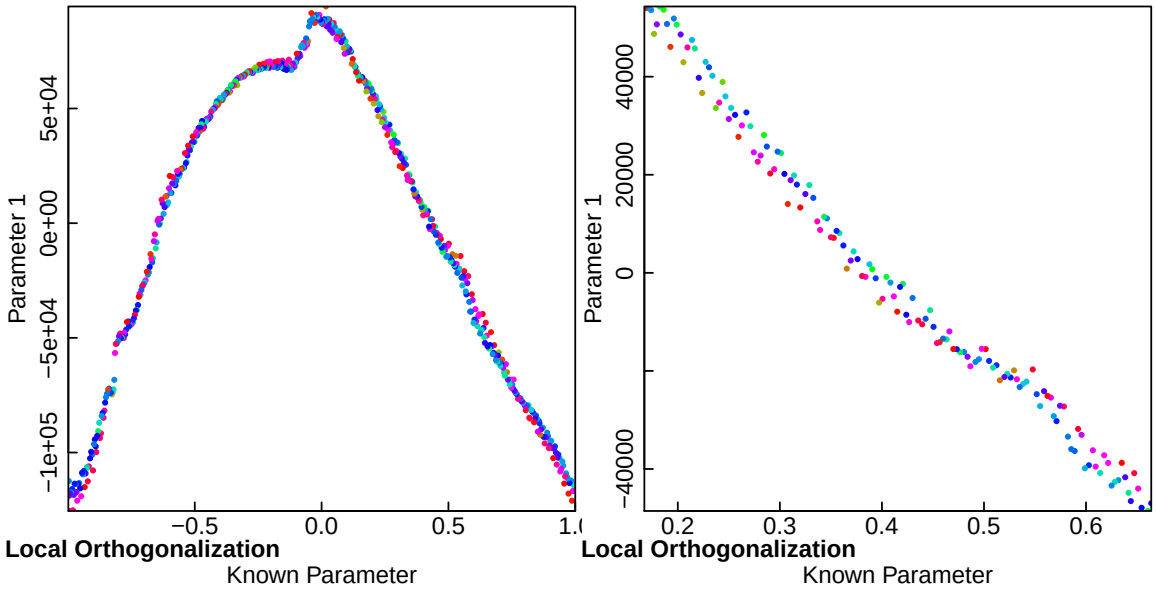



Figure 4.11: Output coordinates for the local orthogonalization method. Left: The local orthogonalization results on the rabbit data set over all angles. The method is unable to deal with a cyclic dataset. The cyclic known parameter is shown on the x-axis within the range $[-1, 1]$, with the left side looping to the right side. Right: The rabbit over $\frac{\pi}{2}$ radians of angle (shown on the x-axis in the range $[0.16, 0.66]$). Due to the systematic bias in computing shortest paths in the neighborhood graph, the output of this method is still highly correlated to the known parameter (although less correlated than standard Isomap).

### 4.6.4 Applying Orthogonalization Locally

Similar to the local distance subtraction method presented in section 4.6.2, we can apply orthogonalization locally within each neighborhood, to create a new distance matrix. Instead of trying to subtract the known distances directly, we use CCA to determine a direction in each neighborhood correlated with the known parameter and remove variation in that direction using orthogonalization.

We begin by computing the neighborhood graph. For each point's neighborhood, we find the direction most correlated with the known parameter. The points are then projected into the subspace not spanned by this direction. Figure 4.10 depicts these steps. The newly created distances should now be independent of the known coordinates within each neighborhood. The new distance matrix can then be converted into a kernel matrix and finally output coordinates in the usual manner. Since distances are only computed for neighbors, the original neighborhood associations must be used.

As a preprocessing step, to remove noise we run PCA on the neighborhood to reduce it to the number of dimensions in the embedding space. This forces the CCA direction to be within the tangent space of the local neighborhood. As a side effect, this solves the tendency of CCA to overfit in high dimensional spaces.

Unfortunately, the created distances are not completely independent of the known parameter (figure 4.11). The main reason for this is that even if each local distance measure is independent of the known parameter, there will be some noise within the measure and this noise accumulates when measuring distances along shortest paths in the neighborhood graph. Since, long distances correspond to paths through the graph with many edges, the accumulation of noise is proportional to the distance between points in the original graph which was dependent on the known parameter. This systematic bias causes the output parameter to be correlated with the known parameter as seen in figure 4.11(right).

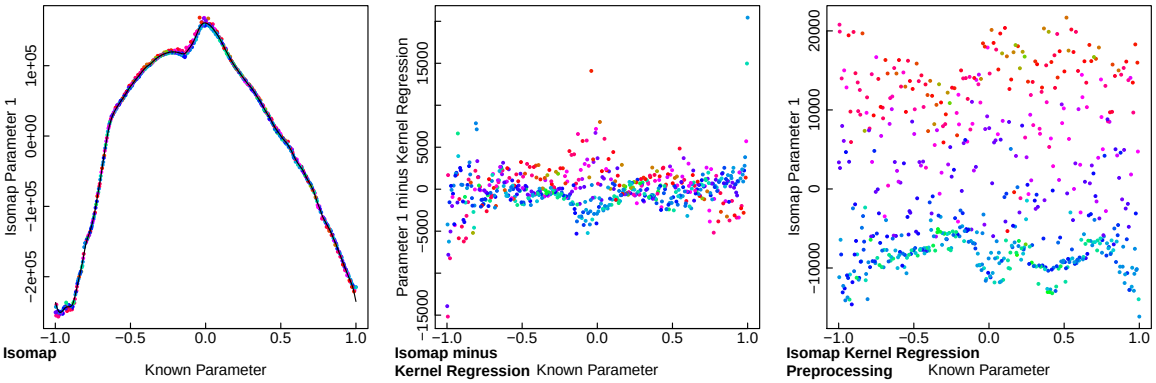



Figure 4.12: Left: The first Isomap parameter of the rabbit over all angles plotted against the known parameter overlaid with a kernel regression curve. Middle: The same Isomap parameter with the kernel regression curve subtracted to highlight variations not due to angle changes. Right: The first Isomap parameter when the rabbit data has been preprocessed by subtracting the weighted local average image in known parameter space from each image.

### 4.6.5 Trend Subtracted Manifold Learning

With orthogonalization, in attempting to decorrelate coordinates from the known parameter, we have essentially modeled the dependence between the known parameter and the manifold using linear and low order polynomial curve segments. Instead of curve segments we focus on using kernel regression to model the dependence between the known parameter and the manifold in a smoother and more adaptable way. A Gaussian kernel is used to compute the weighted average of points in the first Isomap parameter (figure 4.12(left)). This smooth curve is subtracted from the original data to highlight the secondary variation (figure 4.12(middle)). Although not perfect, this method does not suffer from the discontinuities created in windowed orthogonalization and does not require careful parameter setting.

Additionally, instead of performing the kernel regression on the Isomap parameterization, we can perform it on the input data beforehand. This means that we pre-process each image as the original image minus a weighted average of the surrounding images (in the known parameterization) and then perform standard manifold learning on these new images. The width of the kernel used determines which structures in the data will remain. In section 4.7.1 we present a method for choosing an appropriate kernel width.

(a) Data Images (b) Trend Images (c) Trend Subtracted

Figure 4.13: The input images for trend subtracted PUML on two data samples from the kestrel dataset. (a) The original data acquisition images. (b) A Gaussian kernel blurred image in known parameter space around the sample image. For this dataset, this is a time blurred image which captures the general aspect of the bird. (c) The difference between the two images, this image highlights changes in the image which are independent of the known parameter, in this case time. For the top image, this highlights the wing position being in a middle position and the tail being in a slightly lower than average position. For the bottom image, this highlights a wing position slightly higher than average and a tail position slightly lower and wider than average.

After this preprocessing step, the primary variability in the dataset should be independent of the known partial coordinates. If needed, orthogonalization can be used to correct any remaining correlation with the known parameter. We call this method trend subtracted PUML. Of the methods presented in this chapter, this is the most effective.

In summary, this method has the following steps.

1. For each sample image, compute the weighted average of the nearby neighboring images in known parameter space. With a Gaussian kernel, this is effectively a Gaussian blur through known parameter space of the images. This image captures the trend of the image with varying known parameter. Figure 4.13 shows a sample image, its Gaussian kernel blurred equivalent, and their difference for the kestrel dataset.

2. For each sample image, the difference between the blurred trend image and the sample image is computed. These images mainly incorporate variations which are independent of the known parameter.

3. Isomap, or another manifold learning algorithm, is run on these difference images to produce output coordinates. These coordinates should be independent of the known parameter.

4. Optionally, global orthogonalization is run on the output coordinates to produce new coordinates which have zero correlation with the known parameter.

## 4.7 Results

Figures 4.14 through 4.19 show results of running the PUML algorithms on a number of different datasets. In each plot the color of the dots correspond to the ground truth parameterization value. In general, the results of running the distance subtraction methods (both global and local), and the global and local orthogonalization algorithms is very similar to plotting the first Isomap parameter against the known parameter (shown on the left in the second row). The windowed orthogonalization methods frequently finds a different signal within the data, however, it also has the tendency to introduce discontinuities at the edges of the windows. The trend subtraction method works the best of all the presented methods and finds a signal similar to ground truth for all the datasets.

We now describe in more detail the results for each dataset. Figure 4.14 shows results for a synthetic low dimensional manifold: a bent helicoid-like manifold. The known parameter for the helicoid is the oriented distance from the center line, and the ground truth is the distance along the centerline of the helicoid. Isomap itself produces a bent shape, where the first dimension is essentially the distance along the helicoid and the second parameter is the absolute value of the distance to the center line (figure 4.14 upper right). This means that plotting the first Isomap parameter against the known parameter yields a full parameterization of the manifold (figure 4.14 second row, left). All of our methods except the windowed orthogonalization are able to preserve this already good embedding. The windowed orthogonalization methods although still fairly good, have introduced errors along the edges of the windows; thus, producing worse results than the original Isomap parameterization they received as input.

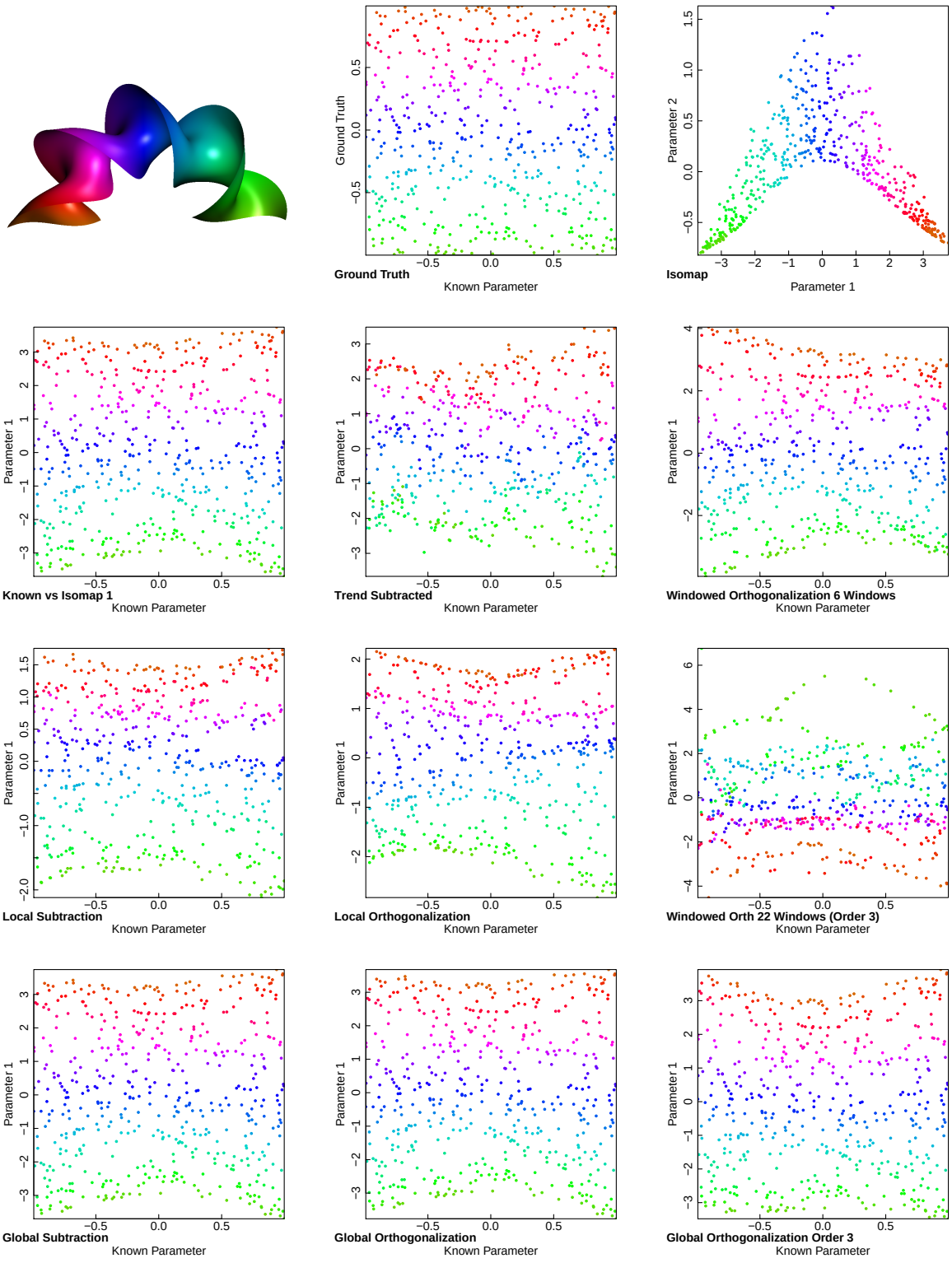



Figure 4.14: PUML methods applied to the bent helicoid-like manifold. Color in the figures corresponds to ground truth position.

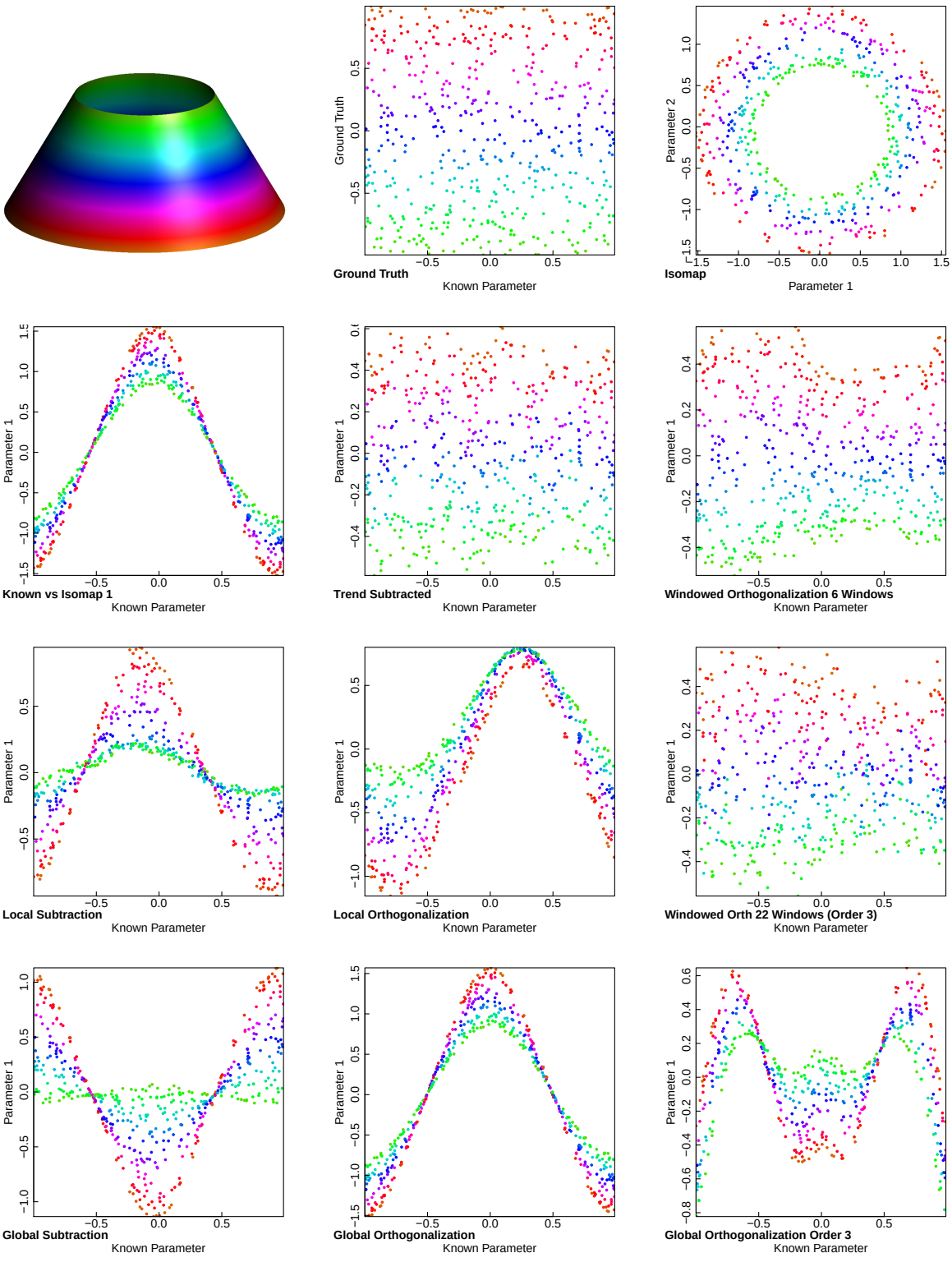



Figure 4.15: PUML methods applied to a conical frustum dataset. Color in the figures corresponds to ground truth position. Notice that several of our methods are able to parameterize the frustum despite its cyclic nature.

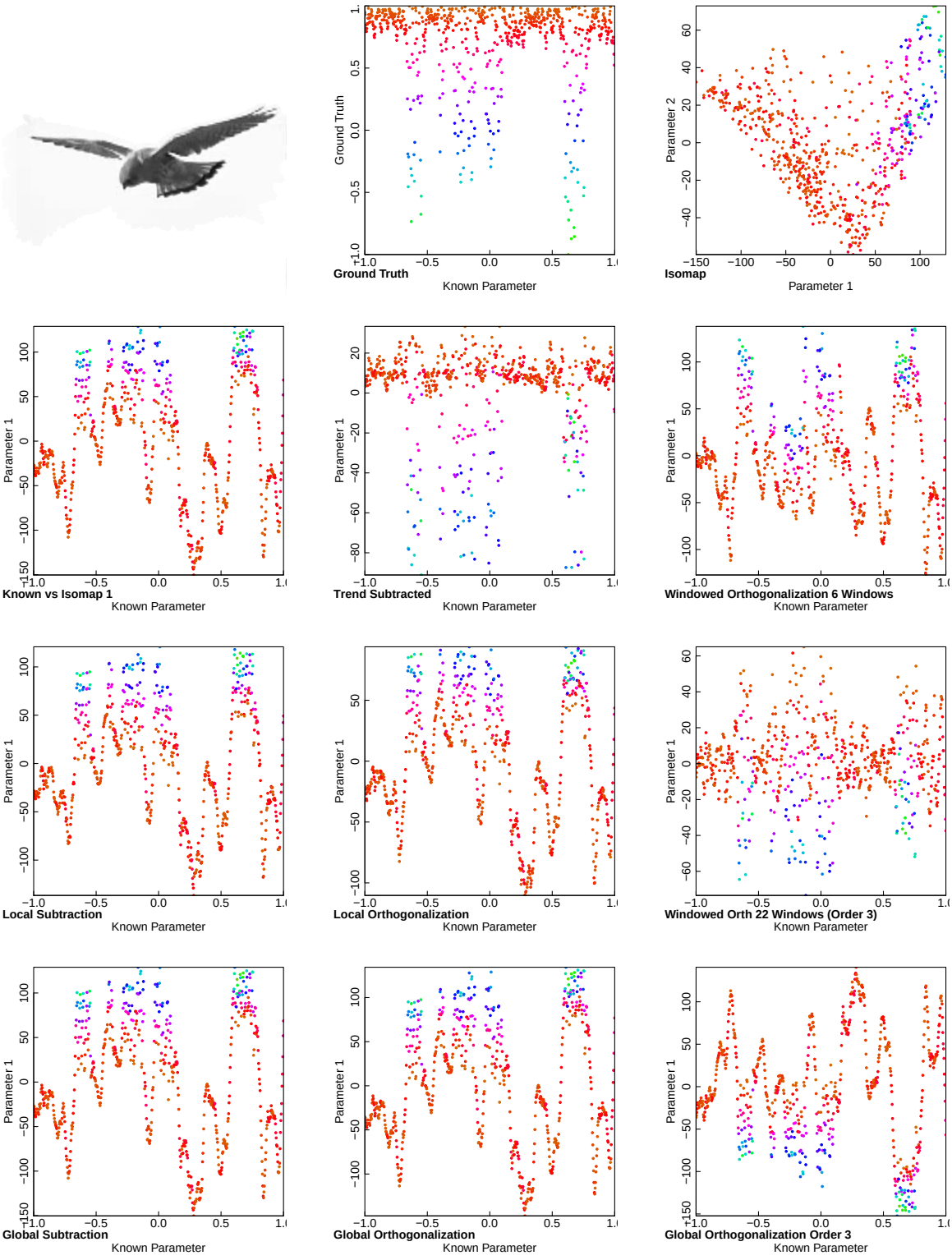



Figure 4.16: PUML methods applied to the kestrel. Color in the figures corresponds to ground truth position.

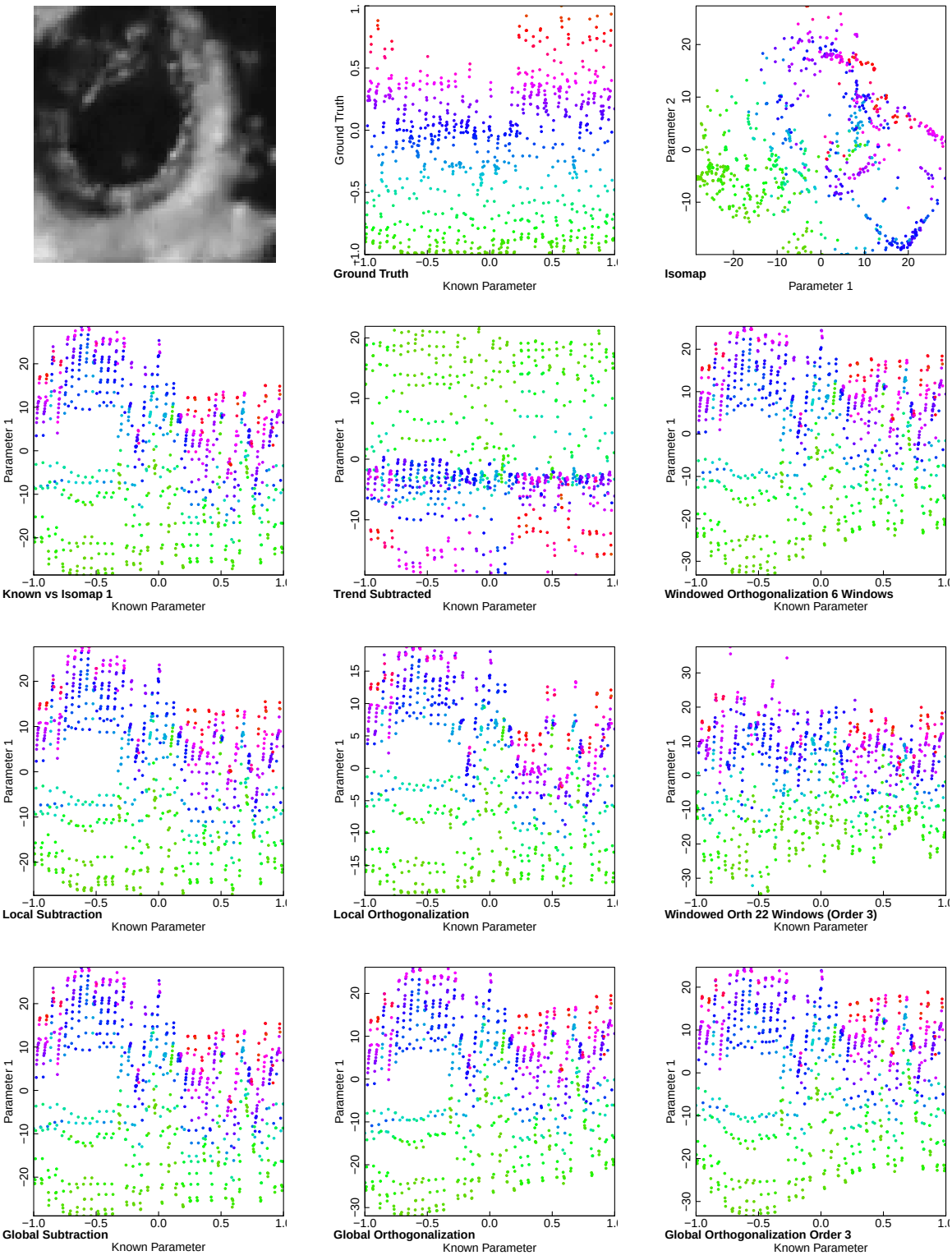



Figure 4.17: PUML methods applied to the heart valve dataset. Color in the figures corresponds to ground truth position.

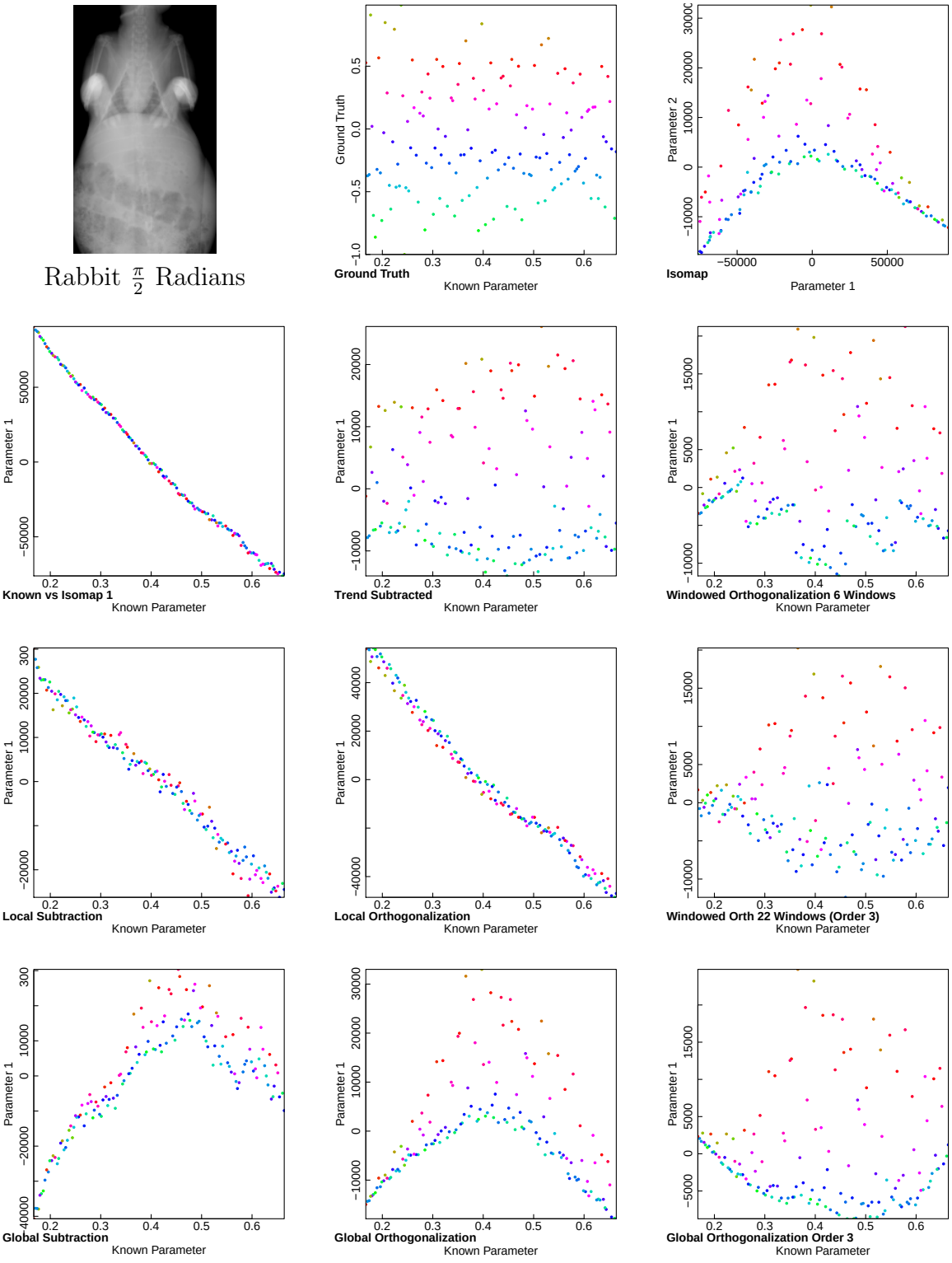



Figure 4.18: PUML methods applied to the Rabbit over $\frac{\pi}{2}$ radians. Color in the figures corresponds to ground truth position.

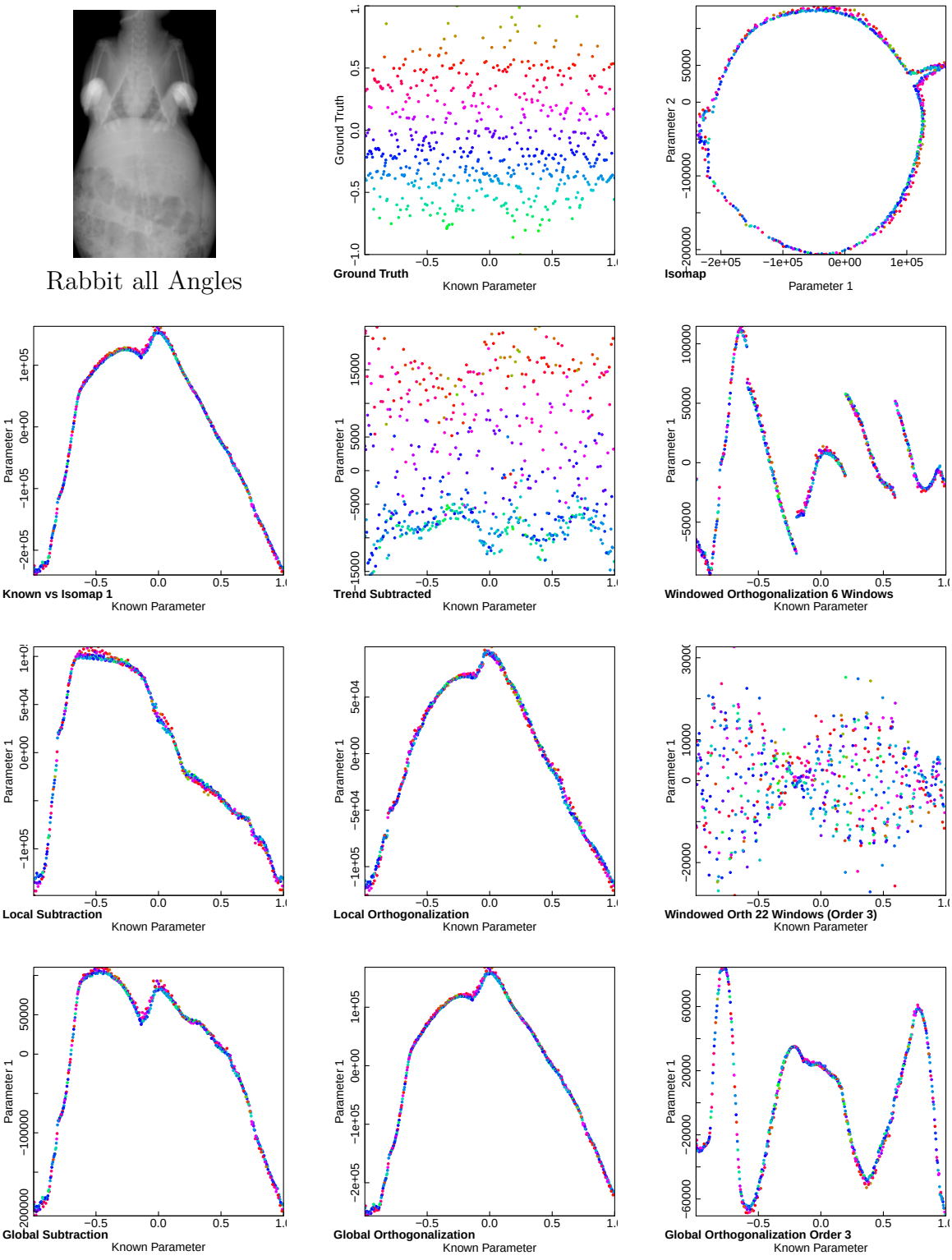



Figure 4.19: PUML methods applied to the Rabbit over all angles. Color in the figures corresponds to ground truth position.

For the conical frustum in figure 4.15 the known parameter is the angular position of the point, while the ground truth value is the height of the point. The known parameter in this case is cyclic. Due to the cyclic nature of the manifold, Isomap produces a ring as the embedding. The first Isomap parameter plotted against the known parameter produces a cosine like curve (second row, left). The distance subtraction methods and local and global orthogonalization methods produce slightly better embeddings by spreading out the points, however, they are far from perfect, as they also display cosine like shapes. The windowed orthogonalization method is able to extract a good embedding by removing the cosine like component in each part of the embedding. Trend subtraction also produces a similarly good embedding by tracking and removing the cosine like component.

Figure 4.16 shows embeddings for the kestrel dataset. For the kestrel dataset, the known parameter is the timestamp of the video frame, while the ground truth is the position of the kestrel's right wing. Isomap finds a signal very related to the ground truth in its first parameter. However, most of the range of the first parameter of Isomap is within the red range of the ground truth, which is a very small portion of the ground truth's range. In plotting the first Isomap parameter against the known parameter, the problem becomes clear, there are very significant long term trends in the data (second row, left). As with the other data sets, distance subtraction and local and global orthogonalization produce very similar embeddings. Windowed orthogonalization produces different results depending on the number of windows and whether higher order orthogonalization is used. Windowed orthogonalization with 6 windows has windows which are too large and contain most of the long term trends within them. This causes little of the long term trend to be removed during orthogonalization. Third order windowed orthogonalization with 22 windows has small enough windows to remove the long term trend. However, windowed orthogonalization centers each window of the dataset around zero, causing red points in the sections of the video with little variation to have the same value as blue points in sections which have lots of variation. Trend subtracted PUML is able to extract a very good signal which removes the long term trends, but does not center the parts of the dataset with lots of variation around zero. This plot is very similar to the ground truth values.

Figure 4.17 shows results for the heart valve image datasets. In this dataset, the known parameter is the timestamp of the video frame and the ground truth value is the width of the heart valve. Isomap produces an embedding where the first parameter is fairly similar to the ground truth; however, as with the kestrel, there is some drift due to long term trends which are visible when plotting the first Isomap coordinate against the known parameter (second row, left). Local and global subtraction and local orthogonalization produce similar embeddings. In this case, global orthogonalization, both first and third order, line up the colors, which correspond to the ground truth values, effectively by raising the values in the second part of the dataset. Windowed orthogonalization with 6 windows has a similar effect, but is not any better, since the windows are too large to remove the long term trend which occurs on a time scale smaller than the window size. Windowed orthogonalization with 22 windows has a small enough window size to remove the long term trends and lines up the ground truth colors well. Trend subtracted PUML is again the most effective method: producing a good embedding which only has problems separating cyan and blue points.

Figure 4.18 shows embeddings for the rabbit over $\frac{\pi}{2}$ radians. The known parameter for this datasets is the angle that the image was acquired from and the ground truth is the diaphragm position of the rabbit. Because the main variation in the dataset is due to changes in the viewing angle, the first Isomap coordinates are almost perfectly correlated with the known parameter. Local and global subtraction and local orthogonalization are unable to extract a signal which is uncorrelated with the known parameter. Global orthogonalization produces an embedding where the ground truth parameter is visible, although it is still confounded with the known parameter. Third order global orthogonalization produces a better embedding where much of the curve has been removed and the ground truth coloring lines up better with the output parameter. Windowed orthogonalization produces an embedding with a jagged bottom edge, where discontinuities appear at the edges of the windows. As with the other datasets, trend subtracted PUML produces the best embedding and is very similar to the ground truth values.

For the rabbit over all angles in figure 4.19, Isomap embeds the cyclic manifold in a ring configuration reminiscent of the embedding of the frustum. Local and global subtraction, and local and global orthogonalization are unable to extract a useful

signal. Windowed orthogonalization with few windows simply produces a discontinuous set of curves, and when lots of windows and third order orthogonalization is used only noise remains. However, trend subtracted PUML is able to produce a good embedding of the dataset, where the only problem is a lack of separation between cyan and blue points.

In summary, trend subtracted PUML is the most effective of our methods and produces useful results for both the synthetic datasets and image datasets. In general, we are able to extract signals from each dataset which are very similar to the ground truth values.

### 4.7.1 Finding the Kernel Width

For trend subtracted PUML, in addition to any free parameters of the manifold learning algorithm, we must specify the correct kernel width for the preprocessing step. A value that is too small will miss the secondary data trend, while one that is too large will incorporate too much of the variation which is dependent on the known parameter. For example in the rabbit data, the kernel width should be large enough to encompass several breaths, but not so large as to incorporate too much variation due to viewing angle.

To determine the optimal kernel width, we look at the correlation between the output parameterization and the ground truth (figure 4.20(top)). On the x-axis we have increasing sizes of kernel width. The value displayed on the x-axis $k$ corresponds to choosing a kernel width such that a weight of 5% of the maximum weight is placed on the $k$th neighbor. On the y-axis the correlation between ground truth and the first parameter using trend subtracted PUML with the specified kernel width is shown. The red horizontal line shows the correlation between the ground truth and the first parameter of standard Isomap.

Unfortunately, the top row of plots would not be available in an application setting since the ground truth is not known, we therefore look at using the variance of the output parameter as a method to set the kernel width. This value is plotted in the bottom row of figure 4.20. In general, the variance increases with increasing kernel

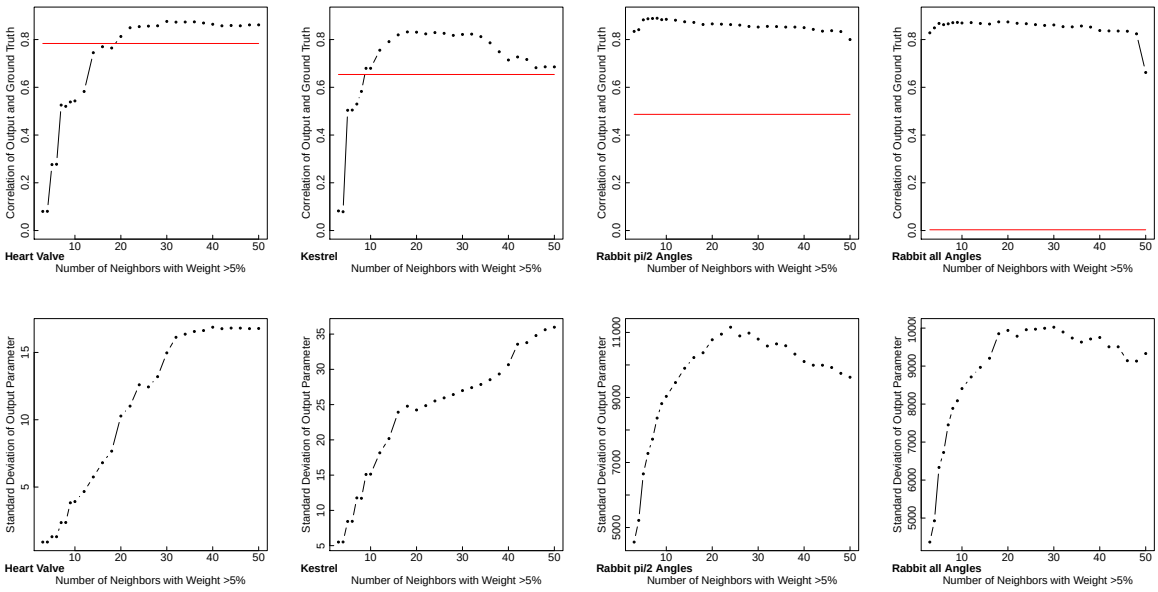



Figure 4.20: Top: The correlation between the parameterization and ground truth. A red horizontal line is drawn at the correlation level between Isomap with no preprocessing and ground truth. This information requires knowing the ground truth, which is unavailable in an application setting. Bottom: Variance of the output parameterization plotted against kernel widths. The kernel width is set to a value such that the $n$th nearest neighbor has a kernel weight of 5%. The knee in these plots generally corresponds to a high correlation to ground truth.

widths, since less of the short term trends are removed with larger kernel widths. We observe that the knee in the output variance plots generally corresponds to a high value of correlation between the output and ground truth. At this kernel width the small scale variance, which is independent of known parameter, is not removed, but the long scale variance, which is dependent on known parameter, is removed.

We consider in more detail each dataset. The first column of figure 4.20 shows results for the heart-valve ultrasound video. This is the simplest of the image datasets, since the dominant visible variation in the scene is the heart valve, and plain Isomap applied directly to the image data correlates well (almost 0.8) with the ground truth measure of the valve opening. The viewpoint drift of the heart is very slow, so using a large kernel width is optimal. The knee of the standard deviation of the output parameter (bottom row, column 1) comes at a rather large value (32). The corresponding value in the correlation image shows that the correlation between the output parameter at this stage and ground truth is quite high, and is above that of standard Isomap.

The second column of figure 4.20 shows results for the kestrel video. Here plain Isomap gives a parameterization which is less correlated (0.65) with the wing flap parameter of interest, because the bird has substantial drift and pose variations through the scene. When the kernel size is chosen around the knee in the standard deviation plot, so that 15-30 neighbors are included in the trend subtraction, the parameterization improves to above 0.8 correlation.

The final two columns of figure 4.20 show results for the partial and full cycle rabbit CT. The parameter of interest is the breathing, but because dramatic image changes are caused by changing the projection angle, plain Isomap is much less correlated. This is especially true over the full cycle (last column), since Isomap is unable to parameterize cyclic variation. The correlation between the trend-subtracted parameterization and the ground truth is maximal when the kernel size includes about 7-9 frames as neighbors. This kernel size includes several breaths: performance declines slowly as the kernel size increases and more of the viewpoint variation is incorporated. Again, the knee of the standard deviation of the output parameter is located at a point where the output is highly correlated to the ground truth.

We are able to automatically choose a good kernel width for each of the data sets. However, for the varied example data we have considered, trend subtracted PUML

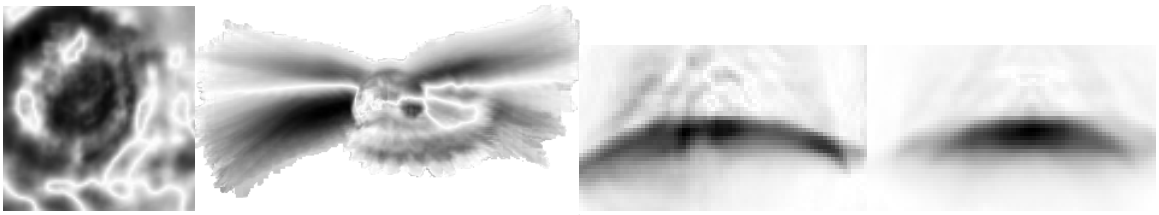
provides good results for a large range of kernel widths, providing evidence that selecting an optimal kernel width is not a delicate process.

### 4.7.2 Correlation Images

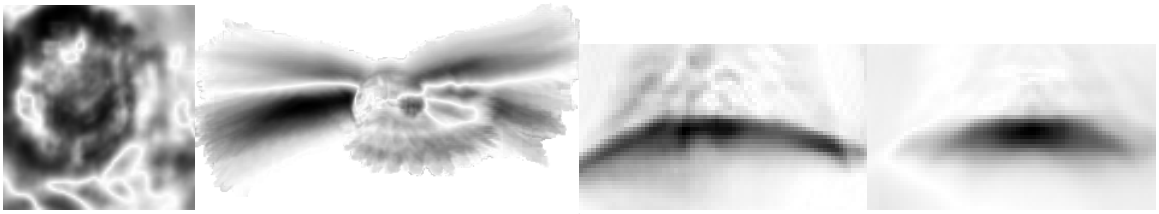
To better understand the image based datasets we look at the correlations between our results and the signal observed at each pixel of the image through time. The intensity of each pixel in figure 4.21 displays the magnitude of the correlation between the signal at that pixel (throughout the video sequence) and various parameters. Darker pixels are more correlated. The first row shows correlation with ground truth, the second with trend subtracted PUML, the third with the first Isomap parameter, the fourth with the second Isomap parameter, and the fifth with the known parameter. The width of the kernel used in trend subtracted PUML is that determined by the knee in figure 4.20(bottom). In general, the images showing correlation with ground truth are very similar to those with trend subtracted PUML.

We consider in more detail each dataset. For the heart valve, the pixels most correlated with ground truth are in the center and along the outside area of the valve, where the motion from opening and closing is most present. This is also the area which is most correlated with the trend subtracted PUML signal, and the first Isomap parameter. The location of pixels correlated with the second Isomap parameter and with the known parameter do not show any clear trend.

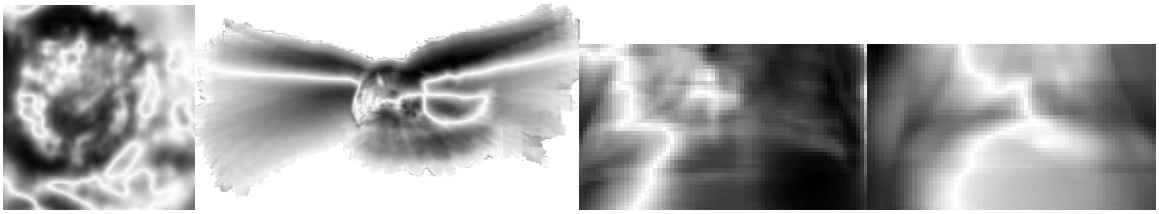
For the kestrel, the wing on the left side of the image is more highly correlated with the ground truth values, since the position of that wing was used to ground truth the dataset. The trend subtracted PUML coordinates are also correlated with the wing positions, although less of the wing in the left side of the image is highly correlated. The first Isomap coordinate seems to be correlated with an upward shift of the entire body of the kestrel. The second Isomap coordinate looks like a Discrete Cosine Transform (DCT) component of the same shift. The known parameter is most correlated with a minuscule shift in the beak and head position, which is the greatest long term trend in the video.



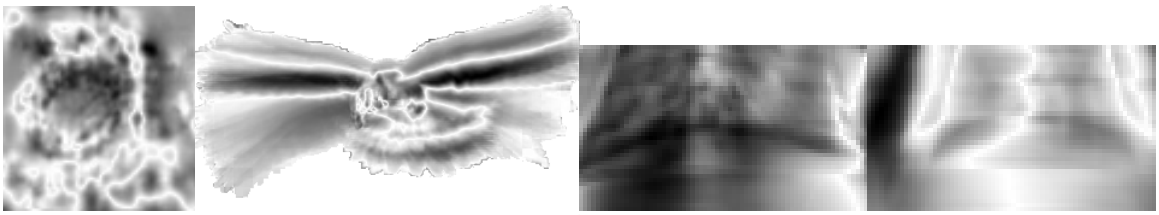
Correlation with Ground Truth



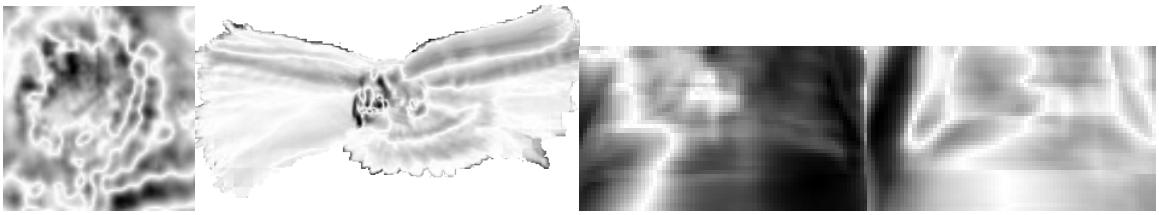
Correlation with Trend Subtracted PUML Results



Correlation with Isomap Parameter 1



Correlation with Isomap Parameter 2



Correlation with Known Parameter

Heart Valve Kestrel Rabbit $\frac{\pi}{2}$ Angles Rabbit all Angles

Figure 4.21: The correlation between pixel locations and from top to bottom: ground truth, first output parameter in trend subtracted PUML, Isomap parameter 1, Isomap parameter 2, and known parameter.

For the rabbit, both the ground truth and trend subtracted PUML both highlight the diaphragm. Isomap component 1 and 2 mainly highlight the rotation of the rabbit; however, the diaphragm is highlighted a little in the second Isomap parameter. The known parameter is strongly correlated with areas of the image which change the most during rotation: the right (and some of the left) over a quarter turn of angles and the left over all angles.

## 4.8 Conclusion

We have posed the problem of Partially Unsupervised Manifold Learning (PUML) in which an embedding of a dataset is found which preserves the structure within the dataset while ensuring that the output parameterization is independent of a given known parameter. We have developed different methods which modify data structures at various stages of the Isomap algorithm to produce a new PUML algorithm. Our methods have included, modifying the pairwise distance matrix, using orthogonalization to ensure output parameterizations are uncorrelated with the known parameter, and removing trends in the dataset found through kernel regression. Using these methods we are able to produce improved parameterizations of a dataset given extra information given in the form of a known parameter.

# Chapter 5

# Conclusion

Finding and characterizing motion patterns and deformation in video is a key to applications ranging from surveillance to medical imaging. In this dissertation we have explored approaches to this problem where the camera is stationary. Because the camera is stationary, it is easier to collect statistics of the observed motion. We characterized motion at the local level: defining motion patterns for each pixel and building this into an understanding of the scene. And, we looked at motion at the global level: characterizing variations over time due to patient breathing, and object movement and deformation. We demonstrate the value of these two approaches by delving deeply into 3 problem domains. For these domains we extended the boundaries of what is possible for fully automated video understanding.

When observing a traffic scene, we developed energy functions to create vector fields from spatio-temporal derivative structure tensors. However, more importantly, we are able to pass the energy functions and the structure tensors to a snake application which can use the energy functions directly. The allows us to robustly create parameterizations of the direction, speed, and width of roads.

In medical imaging, our analysis of motion has hinged on two problems. First, how can the breath phase of a data acquisition automatically be determined? Once the breath phase has been found, how can the entire dataset be synthesized into a coherent model of how the lung moves during breathing? The first question we answered by parameterizing each local area of the lung independently and then aligning each coordinate system to create a consistent breath measure value for every data acquisition. The second question we answered by constructing a motion model based on a single reference volume and a deformation map parameterized by the breath phase. This

model is applicable to more than just lung models, and even generalizes to multiple causes of variation, such as breathing and heartbeat. These are powerful methods for understanding the motion within medical datasets, which can be used for radiation treatment planning and other applications.

Our last problem domain was more theoretical in nature. In Partially Unsupervised Manifold Learning (PUML), a known parameter related to the variation within a dataset is given a priori. The task is to parameterize the variation in the dataset which is not accounted for by the known parameter. We are able to analyze videos and automatically parameterize short term variations while ignoring long term trends in the data. This allows a more coherent analysis of video data which does not suffer from distracting variation within the scene. Additionally, we are able to apply PUML methods to datasets where a known cyclic parameter is present, such as in medical images from a Conebeam CT scanner.

Collectively, these 3 chapters highlight the power of continuing to view the same scene and characterizing the sources of variation within that scene. Automated methods for this will be increasingly important in surveillance and medical imaging as the amount of imagery captured continues to grow much faster than the available manpower to evaluate it.

# References


[1] Wael Abd-Almageed, Christopher E. Smith, and Samah Ramadan. Kernel Snakes: A Non-parametric Active Deformable Model. In *International Conference on Systems, Man and Cybernetics*, October 2003.

[2] Vassili Alexiadis, James Colyar, and John Halkias. A Model Endeavor: A Public-private Partnership is Working to Improve Traffic Microsimulation Technology. *FHWA Public Roads*, 2007.

[3] Saad Ali and Mubarak Shah. A Lagrangian Particle Dynamics Approach for Crowd Flow Segmentation and Stability Analysis. In *CVPR*, 2009.

[4] L. Auslander. On Curvature in Finsler Geometry. *Trans. of the Amer. Math. Soc.*, 79:378–388, 1955.

[5] Simon Baker, Daniel Scharstein, J. P. Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A Database and Evaluation Methodology for Optical Flow. In *ICCV*, pages 1–8, 2007.

[6] M. S. Bartlett. The Statistical Significance of Cononical Correlations. *Biometrika*, 32(1):29–37, 1941.

[7] Saurav Basu, Dipti Prasad Mukherjee, and Scott T. Acton. Implicit Evolution of Open Ended Curves. In *ICIP*, pages 261–264, 2007.

[8] Mirza Faisal Beg, Michael I. Miller, Alain Trouvé, and Laurent Younes. Computing Large Deformation Metric Mappings via Geodesic Flows of Diffeomorphisms. *IJCV*, 61(2):139–157, 2005.

[9] M. Belkin and P. Niyogi. Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering. In *NIPS*, 2002.

[10] M. Bicego, S. Dalfini, G. Vernazza, and V. Murino. Automatic Road Extraction From Aerial Images by Probabilistic Contour Tracking. In *ICIP*, pages 585–588, 2003.

[11] Stephan Bischoff and Leif P. Kobbelt. Parameterization Free Active Contour Models with Topology Control. *Visual Computer*, 20(4):217–228, 2004.

[12] Christopher M. Bishop and Christopher K. I. Williams. GTM: The Generative Topographic Mapping. *Neural Computation*, 10:215–234, 1998.

[13] Matthew Brand. Charting a Manifold. In *NIPS*, pages 961–968. MIT Press, 2003.

[14] A. Bruhn, J. Weickert, C. Feddern, T. Kohlberger, and C. Schnorr. Variational Optical Flow Computation in Real Time. *IEEE Transactions on Image Processing*, 14(5):608–615, May 2005.

[15] A. Brun, C.-F. Westin, M. Herberthson, and H. Knutsson. Fast Manifold Learning Based on Riemannian Normal Coordinates. In *Proceedings of the 14th Scandinavian Conference on Image Analysis (SCIA'05)*, Joensuu, Finland, June 2005.

[16] Matthias Butenuth. Segmentation of Imagery Using Network Snakes. In *Int. Arch. of Photo. and Remote Sens.*, 2006.

[17] R. H. Byrd, P. Lu, and J. Nocedal. A Limited Memory Algorithm for Bound Constrained Optimization. *SIAM Journal on Scientific and Statistical Computing*, 16(5):1190–1208, 1995.

[18] Raghavendra Chandrashekara, Raad Mohiaddin, and Daniel Rueckert. Comparison of Cardiac Motion Fields from Tagged and Untagged MR Images Using Nonrigid Registration. In *FIMH*, pages 425–433, 2005.

[19] Heeyoul Choi and Seungjin Choi. Robust Kernel Isomap. *Pattern Recognition*, 40(3):853–862, 2007.

[20] C. Davatzikos and J. L. Prince. Adaptive Active Contour Algorithms For Extracting and Mapping Thick Curves. In *CVPR*, pages 524–529, June 1993.

[21] B. C. Davis, P. T. Fletcher, E. Bullitt, and S. Joshi. Population Shape Regression From Random Design Data. In *ICCV*, 2007.

[22] Vin de Silva and Mikael Vejdemo-Johansson. Persistent Cohomology and Circular Coordinates. In *Symposium on Computational Geometry (SCG)*, June 2009.

[23] Michael Dixon, Nathan Jacobs, and Robert Pless. Finding Minimal Parameterizations of Cylindrical Image Manifolds. In *CVPR Workshop*, 2006.

[24] David L. Donoho and Carrie Grimes. Hessian Eigenmaps: Locally Linear Embedding Techniques for High-dimensional Data. In *Proc. Natl. Acad. Sci. USA*, May 2003.

[25] David L. Donoho and Carrie Grimes. Local ISOMAP Perfectly Recovers the Underlying Parametrization of Occluded/Lacunary Libraries of Articulated Images. In *CVPR*, June 2003.

[26] David L. Donoho and Carrie Grimes. Image Manifolds Which are Isometric to Euclidean Space. *J. Math. Imaging Vis.*, 23(1):5–24, 2005.

[27] Paul Dupuis, Ulf Grenander, and Michael I. Miller. Variational Problems on Flows of Diffeomorphisms for Image Matching. *Quarterly of Applied Mathematics*, 56:587–600, 1998.

[28] J. Ehrhardt, R. Werner, D. Säring, T. Frenzel, W. Lu, D. Low, and H. Handels. An Optical Flow Based Method For Improved Reconstruction of 4D CT Data Sets Acquired During Free Breathing. *Med. Phys.*, 34(2):711–21, 2007.

[29] Carl Henrik Ek, Philip H. S. Torr, and Neil D. Lawrence. GP-LVM for Data Consolidation. In *Learning from Multiple Sources (NIPS workshop)*, 2008.

[30] Cornelia Fermüller and Robert Pless. The Ouchi Illusion as an Artifact of Biased Flow Estimation. *Vision Research*, 40(1):77–95, 2000.

[31] Cornelia Fermüller, David Shulman, and Yiannis Aloimonos. The Statistics of Optical Flow. *Computer Vision and Image Understanding*, 82(1):1–32, 2001.

[32] M. A. Fischler, J. M. Tenenbaum, and H. C. Wolf. Detection of Roads and Linear Structures in Low-resolution Aerial Imagery Using a Multisource Knowledge Integration Technique. *Readings in computer vision: issues, problems, principles, and paradigms*, pages 741–752, 1987.

[33] Daniel Freedman. Efficient Simplicial Reconstructions of Manifolds From Their Samples. *PAMI*, 24(10):1349–1357, October 2002.

[34] Andreas Geiger, Raquel Urtasun, and Trevor Darrell. Rank Priors for Continuous Non-Linear Dimensionality Reduction. In *CVPR*, June 2009.

[35] Manfred Georg, Jonathon J Cannon, Andrew J Hope, Wei Lu, Daniel A Low, and Robert B Pless. Automating 4D CT Reconstruction Using Manifold Learning. In *American Radium Society Annual Meeting*, May 2007.

[36] Manfred Georg and Robert Pless. Fitting Parametric Road Models to Spatiotemporal Derivatives. In *Machine Learning for Vision-based Motion Analysis (ICCV Workshop)*, September 2009.

[37] Manfred Georg, Richard Souvenir, Andrew Hope, and Robert Pless. Manifold Learning for 4D CT Reconstruction of the Lung. In *Workshop on Mathematical Methods in Biomedical Image Analysis (CVPR Workshop)*, June 2008.

[38] Thomas Guerrero, Geoffrey Zhang, Tzung-Chi Huang, and Kang-Ping Lin. Intrathoracic Tumour Motion Estimation From CT Imaging Using the 3D Optical Flow Method. *Physics in Medicine and Biology*, 49:4147–4161, August 2004.

[39] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality Reduction by Learning an Invariant Mapping. In *CVPR*, 2006.

[40] H. Handels, R. Werner, T. Frenzel, D. Säring, W. Lu, D. Low, and J. Ehrhardt. Generation of 4D CT Image Data and Analysis of Lung Tumour Mobility During the Breathing Cycle. *Stud Health Technol Inform*, 124, 2006.

[41] Andrew J Hope, Manfred Georg, Jonathon J Cannon, J Hubenschmidt, Wei Lu, Daniel A Low, and Robert B Pless. Applications of Manifold Learning Techniques in 4D-CT reconstruction. In *International Conference on the use of Computers in Radiation Therapy*, June 2007. Reviewer's Choice.

[42] Berthold K. P. Horn and Brian G. Schunck. Determining Optical Flow. *Artificial Intelligence*, 17:185–203, 1981.

[43] Min Hu, Saad Ali, and Mubarak Shah. Learning Motion Patterns in Crowded Scenes Using Motion Flow Field. In *ICPR*, 2008.

[44] Jiantao Huang, Dana Abendschein, Victor G. Dávila-Román, and Amir A. Amini. Spatio-Temporal Tracking of Myocardial Deformation with a 4D B-Spline Model from Tagged MRI. *IEEE Trans. Med. Imaging*, 18(10):957–972, 1999.

[45] S. Van Huffel and J. Vandewalle. *The Total Least Squares Problem: Computational Aspects and Analysis*. Society for Industrial and Applied Mathematics, Philadelphia, 1991.

[46] M. Kass, A. Witkin, and D. Terzopoulos. Snakes: Active Contour Models. *IJCV*, 1(4):321–331, 1988.

[47] Teuvo Kohonen. *Self-Organizing Maps*. 1990.

[48] L. Kratz and K. Nishino. Anomaly Detection in Extremely Crowded Scenes Using Spatio-temporal Pattern Models. In *CVPR*, pages 1446–1453, 2009.

[49] Neil D. Lawrence. Gaussian Process Latent Variable Models for Visualisation of High Dimensional Data. In *NIPS*, 2003.

[50] Neil D. Lawrence. Probabilistic Non-linear Principal Component Analysis with Gaussian Process Latent Variable Models. *Journal of Machine Learning Research*, 2005.

[51] John Aldo Lee and Michel Verleysen. How to Project Circular Manifolds Using Geodesic Distances? In *European Symposium on Artificial Neural Networks*, 2004.

[52] Gayle Leen and Colin Fyfe. A Gaussian Process Latent Variable Model formulation of Canonical Correlation Analysis. In *European Symposium on Artificial Neural Network (ESANN)*, 2006.

[53] Guang Li, Deborah Citrin, Kevin Camphausen, Boris Mueller, Chandra Burman, Borys Mychalczak, Robert W. Miller, and Yulin Song. Advances in 4D Medical Imaging and 4D Radiation Therapy. *TCRT*, 7(1):67–82, February 2008.

[54] Jian Li, Shaogang Gong, and Tao Xiang. Scene Segmentation for Behaviour Correlation. In *ECCV*, 2008.

[55] Tony Lin, Hongbin Zha, and Sang Uk Lee. Riemannian Manifold Learning for Nonlinear Dimensionality Reduction. In *ECCV*, 2006.

[56] Zhengdong Lu, Prateek Jain, and Inderjit S. Dhillon. Geometry-aware Metric Learning. In *ICML*, June 2009.

[57] Ramesh Marikhu, Matthew N. Dailey, Stanislav S. Makhanov, and Kiyoshi Honda. A Family of Quadratic Snakes for Road Extraction. In *ACCV*, pages 85–94, 2007.

[58] Jamie R. McClelland, Jane M. Blackall, Sǵolnè Tarte, Adam C. Chandler, Simon Hughes, Shahreen Ahmad, David B. Landau, and David J. Hawkes. A Continuous 4D Motion Model From Multiple Respiratory Cycles For Use in Lung Radiotherapy. *Medical Physics*, 33(9):3348–3358, September 2006.

[59] John Melonakos, Eric Pichon, Sigurd Angenent, and Allen Tannenbaum. Finsler Active Contours. *PAMI*, 30(3):412–423, 2008.

[60] Michael I. Miller, Alain Trouvé, and Laurent Younes. On the Metrics and Euler-Lagrange Equations of Computational Anatomy. *Annual Review Biomedical Engineering*, 4:375–405, August 2002.

[61] Antoine Mittal and Nikos Paragios. Motion-Based Background Subtraction using Adaptive Kernel Density Estimation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 302–309, 2004.

[62] Antoine Monnet, Anurag Mittal, Nikos Paragios, and Visvanathan Ramesh. Background Modeling and Subtraction of Dynamic Scenes. In *Proc. IEEE International Conference on Computer Vision*, pages 1305–1312, 2003.

[63] A. Mukherjee, S. K. Parui, d. Chaudhuri, B. B. Chauduri, and R. Krishnan. An Efficient Algorithm for Detection of Road-Like Structures in Satellite Images. In *ICPR*, pages 875–879, 1996.

[64] H.-H. Nagel and M. Haag. Bias-Corrected Optical Flow Estimation for Road Vehicle Tracking. In *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, page 1006, Washington, DC, USA, 1998. IEEE Computer Society.

[65] Summary Report, NGSIM Peachtree Street (Atlanta) Data Analysis. http://www.ngsim.fhwa.dot.gov/, June 2007.

[66] T. Nielsen, R. Manzke, R. Proksa, and M. Grass. Cardiac Cone-beam CT Volume Reconstruction using ART. *Medical Physics*, 32(4):851–860, April 2005.

[67] Stanley Osher and James A. Sethian. Fronts Propagating With Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations. *J. Comp. Phys.*, 79(1):12–49, 1988.

[68] Robert Pless. Detecting Roads in Stabilized Video with the Spatio-Temporal Structure Tensor. *GeoInformatica*, 10(1):37–53, 2006.

[69] Robert Pless, John Larson, Scott Siebers, and Ben Westover. Evaluation of Local Models of Dynamic Backgrounds. In *CVPR*, volume 2, pages 73–78, 2003.

[70] Robert Pless and Ian Simon. Embedding Images in Non-flat Spaces. Technical Report WU-CS-01-43, Washington University in St. Louis, December 2001.

[71] Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Nearest Neighbors in High-Dimensional Data: The Emergence and Influence of Hubs. In *ICML*, June 2009.

[72] Marie Rochery, Ian H. Jermyn, and Josiane Zerubia. Higher Order Active Contours. *IJCV*, 69(1):27–42, 2006.

[73] Sam T Roweis and Lawrence K Saul. Nonlinear Dimensionality Reduction by Locally Linear Embedding. *Science*, 290(5500):2323–2326, December 2000.

[74] Sam T Roweis, Lawrence K Saul, and Geoffrey E Hinton. Global Coordination of Local Linear Models. In *NIPS*, 2001.

[75] David Sarrut, Vlad Boldea, Serge Miguet, and Chantal Ginestet. Simulation of Four-Dimensional CT Images From Deformable Registration Between Inhale and Exhale Breath-hold CT Scans. *Medical Physics*, 33(3):605–617, March 2006.

[76] Ryan Schmidt, Cindy Grimm, and Brian Wyvill. Interactive Decal Compositing with Discrete Exponential Maps. In *SIGGRAPH*, 2006.

[77] Bernhard Scholköpf, Alexander Smola, and Klaus-Robert Müller. Nonlinear Component Analysis as a Kernel Eigenvalue Problem. *Neural Computation*, 10:1299–1319, 1998.

[78] E. Schreibmann, G.T.Y. Chen, and L. Xing. Image Interpolation in 4D CT using a BSpline Deformable Registration Model. *Int J Radiat Oncol Biol Phys*, 2006.

[79] Steven M. Seitz and Simon Baker. Filter Flow. In *ICCV*, 2009.

[80] Amnon Shashua, Anat Levin, and Shai Avidan. Manifold Pursuit: A New Approach to Appearance Based Recognition. In *ICPR*, pages 590–594, 2002.

[81] Le Song, Alex Smola, Karsten Borgwardt, and Arthur Gretton. Colored Maximum Variance Unfolding. In *NIPS*, 2007.

[82] Richard Souvenir, Qilong Zhang, and Robert Pless. Image Manifold Interpolation using Free-Form Deformations. In *ICIP*, pages 1437–1440, October 2006.

[83] M. Tavakoli and R.K. Bajcsy. Computer Recognition of Roads from Satellite Pictures. In *ICPR*, pages 190–194, 1974.

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

[85] N.J. Tustison, V.G. Dvila-Romn, and A.A. Amini. Myocardial Kinematics from Tagged MRI Based on a 4-D B-spline Model. *IEEE Trans Biomed Eng*, 50(8):1038–40, 2003.

[86] Raquel Urtasun, David J. Fleet, Andreas Geiger, Jovan Popović, Trevor J. Darrell, and Neil D. Lawrence. Topologically-Constrained Latent Variable Models. In *ICML*, July 2008.

[87] Ylanga G. van der Geld, Suresh Senan, John R. van Sörnsen de Koste, Harm van Tinteren, Ben J. Slotman, René W. M. Underberg, and Frank J. Lagerwaard. Evaluating Mobility for Radiotherapy Planning of Lung Tumors: A Comparison of Virtual Fluoroscopy and 4DCT. *Lung Cancer*, 53(1):31–37, July 2006.

[88] L. J. P. van der Maaten, E. O. Postma, and H. J. van den Herik. Dimensionality Reduction: A Comparative Review. Published online., 2007. submitted to JMLR 2009.

[89] S. S. Vedam, P. J. Keall, V. R. Kini, H. Mostafavi, H. P. Shukla, and R. Mohan. Acquiring a Four-Dimensional Computed Tomography Dataset Using an External Respiratory Signal. *Physics in Medicine and Biology*, 48(1):45–62, January 2003.

[90] Fernando A. Velasco and José L. Marroquín. Robust Parametric Active Contours: The Sandwich Snakes. *MV&A*, 12(5):238–242, 2001.

[91] Jakob J. Verbeek, Sam T. Roweis, and Nikos Vlassis. Non-linear CCA and PCA by Alignment of Local Models. In *NIPS*, volume 16, pages 297–304. MIT Press, 2003.

[92] Ulrike von Luxburg. A Tutorial on Spectral Clustering. *Statistics and Computing*, 17(4), December 2007.

[93] S. Wang, Y. Markandey, and A. Reid. Total Least Squares Fitting Spatiotemporal Derivatives to Smooth Optical Flow Fields. In *Proc. of the SPIE: Signal and Data Processing of Small Targets*, volume 1698, pages 42–55. SPIE, 1992.

[94] Xiaogang Wang, Keng Teck Ma, Gee-Wah Ng, and Eric Grimson. Trajectory Analysis and Semantic Region Modeling Using a Nonparametric Bayesian Model. Technical Report MIT-CSAIL-TR-2008-015, Massachusetts Institute of Technology, 2008.

[95] Xiaogang Wang, Xiaoxu Ma, and Eric Grimson. Unsupervised Activity Perception by Hierarchical Bayesian Models. In *CVPR*, 2007.

[96] K. Q. Weinberger and L. K. Saul. Unsupervised Learning of Image Manifolds by Semidefinite Programming. In *CVPR*, pages 988–995, 2004.

[97] K. Q. Weinberger and L. K. Saul. An Introduction to Nonlinear Dimensionality Reduction by Maximum Variance Unfolding. In *AAAI*, 2006. Nectar paper.

[98] C Wiedemann and S Hinz. Automatic Extraction and Evaluation of Road Networks from Satellite. In *Int. Arch. of Photo. and Remote Sens.*, 1999.

[99] J. W. Wong, M. B. Sharpe, D. A. Jaffray, V. R. Kini, J. M. Robertson, J. S. Stromberg, and A. A. Martinez. The Use of Active Breathing Control (ABC) to Reduce Margin for Breathing Motion. *International Journal Radiation Oncolology Biology Physics*, 44(4):911–919, July 1999.

[100] John Wright and Robert Pless. Analysis of Persistent Motion Patterns Using the 3D Structure Tensor. In *WACV/MOTION*, pages 14–19, 2005.

[101] Chengyang Xu and Jerry L Prince. Gradient Vector Flow: A New External Force for Snakes. In *CVPR*, pages 66–71, June 1997.

[102] Sheng Xu, Russell H. Taylor, Gabor Fichtinger, and Kevin Cleary. Lung Deformation Estimation and Four-dimensional CT Lung Reconstruction. *Academic Radiology*, 13(9):1082–1092, September 2006.

[103] Y. Yagi, M. Brady, Y. Kawasaki, and M. Yachida. Active Contour Road Model for Smart Vehicle. *ICPR*, 3:811–814, 2000.

[104] Qilong Zhang, Richard Souvenir, and Robert Pless. On Manifold Structure of Cardiac MRI Data: Application to Segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, 1:1092–1098, 2006.

[105] Zhenyue Zhang and Hongyuan Zha. Principal Manifolds and Nonlinear Dimensionality Reduction via Tangent Space Alignment. *SIAM Journal on Scientific Computing*, 2001.

# Vita

Manfred Georg

**Date of Birth** July 31, 1982

**Place of Birth** Bonn, Germany

**Degrees** B.S. Magna Cum Laude, Mathematics and Computer Science, December 2003
M.S. Computer Science and Engineering, December 2007
Ph.D. Computer Science and Engineering, August 2010

**Publications** Manfred Georg and Sergey Gorinsky. Congestion Control with a Misbehaving Receiver: Robust TFRC and Other Protocols. *Journal of Internet Engineering, 3(1), December 2009.*

*Manfred Georg and Robert Pless. Fitting Parametric Road Models to Spatio-temporal Derivatives. In* Machine Learning for Vision-based Motion Analysis (ICCV Workshop)*, September 2009.*

*Manfred Georg, Richard Souvenir, Andrew Hope, and Robert Pless. Manifold Learning for 4D CT Reconstruction of the Lung. In* Workshop on Mathematical Methods in Biomedical Image Analysis (CVPR Workshop)*, June 2008.*

*Manfred Georg, Richard Souvenir, Andrew Hope, and Robert Pless. Simultaneous Data Volume Reconstruction and Pose Estimation from Slice Samples. In* Computer Vision and Pattern Recognition*, June 2008.*

*Sergey Gorinsky, Manfred Georg, Maxim Podlesny, and Christoph Jechlitschek. A Theory of Load Adjustments and its Implications for Congestion Control.* Journal of Internet Engineering, 1(2):82–93, 2007.

Manfred Georg, Christoph Jechlitschek, and Sergey Gorinsky. Improving Individual Flow Performance with Multiple Queue Fair Queuing. In *International Workshop on Quality of Service*, June 2007.

Andrew J Hope, Manfred Georg, Jonathon J Cannon, J Hubenschmidt, Wei Lu, Daniel A Low, and Robert B Pless. Applications of Manifold Learning Techniques in 4D-CT reconstruction. In *International Conference on the use of Computers in Radiation Therapy*, June 2007. Reviewer's Choice.

Manfred Georg, Jonathon J Cannon, Andrew J Hope, Wei Lu, Daniel A Low, and Robert B Pless. Automating 4D CT Reconstruction Using Manifold Learning. In *American Radium Society Annual Meeting*, May 2007.

C. Stringfellow, C.D. Amory, D. Potnurri, M. Georg, and A. Andrews. Comparison of Software Architecture Reverse Engineering Methods. *Journal of Information and Software Technology, (7):484–497, 2006.*

*Manfred Georg and Sergey Gorinsky. Protecting TFRC from a Selfish Receiver. In* Proceedings of the IEEE International Conference on Networking and Services*, October 2005.*

*C. Stringfellow, C.D. Amory, D. Potnurri, M. Georg, and A. Andrews. Deriving Change Architectures from RCS History. In* Proceedings of the IASTED International Conference on Software Engineering and Applications*, November 2004.*

*Horst Hahn and Manfred Georg. Fractal Aspects of Global and Local Optimization Schemes in Constrained Construction of Three-Dimensional Vascular Systems. In* Proceedings of the Fractals in Biology and Medicine Conference*, March 2004.*

*Manfred Georg, Tobias Preusser, and Horst K. Hahn. Global Constructive Optimization of Vascular Systems. Technical Report WUCSE-2010-11, Department of Computer Science*

*and Engineering at Washington University in St. Louis, March 2010.*

*Manfred Georg, Christoph Jechlitschek, and Sergey Gorinsky. Improving Individual Flow Performance with Multiple Queue Fair Queuing. Technical Report WUCSE-2007-30, Department of Computer Science and Engineering at Washington University in St. Louis, May 2007.*

*Sergey Gorinsky, Manfred Georg, Maxim Podlesney, and Christoph Jechlitschek. A Theory of Load Adjustments and Its Implications for Congestion Control. Technical Report WUCSE-2006-40, Department of Computer Science and Engineering at Washington University in St. Louis, August 2006.*

August 2010