UNIVERSITY OF CALIFORNIA, SAN DIEGO

Manifold Learning Techniques For Non-Rigid Structure From Motion

A dissertation submitted in partial satisfaction of the requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Vincent C. Rabaud

Committee in charge:

Professor Serge Belongie, Chair
Professor Sanjoy Dasgupta
Professor David Kriegman
Professor Gert Lanckriet
Professor Truong Nguyen

2009

DEDICATION

This document is dedicated to my family and the Ecole Polytechnique.

EPIGRAPH

If your eye were more acute you would see everything in motion: just like ignited paper becomes deformed, every thing unfolds and deforms.

– Friedrich Nietzsche, Posthumous Fragments.

Then all motion, of whatever nature, creates ?

– Edgar Allan Poe, *The Power of Words*, 1945.

Bust a move

– Young MC.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First I would like to thank my advisor, Serge Belongie, who is the main reason why I decided to do a PhD in computer vision. I always had a special interaction with images but only for leisure. Serge is a passionate person that can communicate his enthusiasm very easily and he made me aware of how exciting computer vision research can be. I also really appreciated the trust and freedom he gave me over the years; we did not believe the ideas developed in this dissertation could ever work but Serge let me play with them and follow my instincts. Serge was also a great personal advisor over the years, which made me realize how much research and personal life are linked.

I would also like to acknowledge Sanjoy Dasgupta for the inspiration he provided me. While our interactions were definitely not as frequent as I would have liked (mostly because my research was not using much machine learning), I still remember his words about the importance of research over classes, ironically enough given during one of his classes. I am also grateful to him for being able to explain anything in the simplest fashion, thus providing me the hope that I could understand anything as long as I could find the right manual/person to provide explanations.

These years at UCSD were also an occasion to work with people from really different backgrounds, and therefore, ways of thinking. Piotr Dollár introduced me to machine-learning and made me approach and think about problems differently. He also made me adopt a better scientific approach: rigorous, open and generous towards others. Kristin Branson was also a huge influence on the development of my scientific mind: she made me appreciate and believe in probabilistic approaches ! She is also an extremely kind person who has always been there to talk about any subject. Sameer Agarwal had the same influence: his exhaustive knowledge always made him the first person to talk to about a new problem I had. I wish I had started a year earlier so that we could have worked together on structure from motion. Manmohan Chandraker also taught me to appreciate another mathematical field: I now agree, the answer to life, the universe, and everything is optimization, not 42.

I also would like to acknowledge the “graphics” people who I shared an office with: they introduced me to efficient and new programming techniques. Thank you Wojciech Jarosz, Neel Joshi, Will Chang and mostly Craig Donner. A special thanks to Neil Alldrin for being my conference buddy and a good friend: too bad I do not study your field. Finally, I would like to thank the older crowd that I barely met for sharing their wisdom: Lawrence Cayton, Satya Mallick, Ben Ochoa, Matt Tong, Eric Wiewora and Josh Wills. Similarly, I wish good luck to the younger crowd ! Daniel Hsu, Boris Babenko, Steve Branson, Carolina Galleguillos and Kai Wang.

I would also like to thank the following grants which supported my research over the years: the NSF Career Grant #0448615, the Alfred P. Sloan Research Fellowship and the UCSD Division of Calit2.

While a great effort has been put on optimization in this thesis, I also went through hours of trials which would not have been possible without the computer grid UCSD FWGrid Project (NSF Research Infrastructure Grant Number EIA-0303622) that provided me with a total 17 years worth of CPU processing ...

Finally, I would like to acknowledge my family. They provided me with support, comfort and love. My mother Catherine, father Hervé, sister Blandine, and girlfriend Brittany McLaren. I owe everything to them.

Chapter 2, is a reprint of the material as it appears in Linear Embeddings in Non-Rigid Structure from Motion, V. Rabaud and S. Belongie, IEEE Conference on Computer Vision and Pattern Recognition, 2009.

Chapter 3, is a reprint of the material as it appears in Non-Isometric Manifold Learning: Analysis and an Algorithm, P. Dollár, V. Rabaud and S. Belongie, International Conference on Machine Learning, 2007 and Learning to Traverse Image Manifolds, P. Dollár, V. Rabaud and S. Belongie, Neural Information Processing Systems, 2006.

Chapter 4, is a reprint of the material as it appears in Re-Thinking Non-Rigid Structure from Motion, V. Rabaud and S. Belongie, IEEE Conference on Computer Vision and Pattern Recognition, 2008.

VITA

1979	Born, Nantes, France
2001	B.S./M.S. Ecole Polytechnique, Paris, France
2003	M.S. SUPAERO, Toulouse, France
2009	Ph.D. UCSD, California, USA

PUBLICATIONS

P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features," *IEEE Transaction on Pattern Analysis and Machine Intelligence* (**PAMI, in preparation**), 2008.

V. Rabaud and S. Belongie, "Shape Embeddings and Non-Rigid Structure From Motion" , *IEEE Conference on Computer Vision and Pattern Recognition*, (**CVPR**), 2009.

V. Rabaud and S. Belongie, "Re-Thinking Non-Rigid Structure From Motion" , *IEEE Conference on Computer Vision and Pattern Recognition*, (**CVPR**), 2008.

S. Steinbach, V. Rabaud and S. Belongie, "Soylent Grid: it's made of People !" , *Interactive Computer Vision, in conjunction with ICCV*, (**ICV**), 2007.

P. Dollár, V. Rabaud and S. Belongie, "Non-Isometric Manifold Learning: Analysis and an Algorithm", *International Conference on Machine Learning*, (**ICML**), 2007.

P. Dollár, V. Rabaud and S. Belongie, "Learning to Traverse Image Manifolds" , *Neural Information Processing Systems*, (**NIPS**), 2006.

V. Rabaud and S. Belongie, "Counting Crowded Moving Objects,", *IEEE Conference on Computer Vision and Pattern Recognition*, (**CVPR**), 2006, pp. 705- 711, vol. 1.

P. Dollár, V. Rabaud, G. Cottrell and S. Belongie, "Behavior Recognition via Sparse Spatio-Temporal Features," *Joint International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, (**VS-PETS**), 2005.

S. Belongie, K. Branson, P. Dollár, and V. Rabaud, "Monitoring Animal Behavior in the Smart Vivarium," *International Conference on Methods and Techniques in Behavioral Research*, 2005.

V. Rabaud and S. Belongie, "Big Little Icons," *IEEE Workshop on Computer Vision Applications for the Visually Impaired, in conjunction with CVPR*, (**CVAVI**), 2005.

K. Branson, V. Rabaud and S. Belongie, “Three Brown Mice: See How They Run,” *Joint International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, (**VSPETS**), 2003, pp. 78-85.

V. Rabaud and B. Deguine “A Geometrical Approach To Determine Blackout Windows At Launch,” AAS/AIAA Space Flight Mechanics Meeting, Ponce, Puerto Rico, (**AAS**), 2003, 03-187

FIELDS OF STUDY

Major Field: Structure from Motion
Rigid Structure from Motion, Non-Rigid Structure from Motion, Image Panoramas.

Major Field: Motion Analysis
Motion Segmentation, Blob Tracking, Real-Time Tracking, Human Motion Analysis.

Major Field: Manifold Learning
Manifold Tangent Learning, Embedding Technique Comparison, Image Manifold Learning.

Minor Fields
CAPTCHAs, Visual Perceptions.

ABSTRACT OF THE DISSERTATION

Manifold Learning Techniques For Non-Rigid Structure From Motion

by

Vincent C. Rabaud
Doctor of Philosophy in Computer Science

University of California, San Diego, 2009

Professor Serge Belongie, Chair

This dissertation establishes the foundations for treating non-rigid structure from motion as a manifold learning problem. Non-rigid structure from motion (or NRSFM) is the computer vision technique that recovers the 3D structure of a deformable object from a monocular video sequence. State of the art techniques assume that the object is generated from a linear combination of basis shapes and several techniques (like EM or matrix decomposition) are then used to exploit this assumption.

So far, nobody has ever considered this setup as a linear manifold learning problem. In this dissertation, we show how NRSFM can be formulated as a manifold embedding recovery problem. First, the different frames of the video sequence are embedded in a low dimensional manifold, from which the shape coordinates are discovered and then used to recover the camera parameters and the shape basis.

We also push the limits of this model by relaxing the linearity assumption of the manifold and only constraining its dimensionality. This looser assumption is proven to be much more valid than the simple linearity one for certain objects where only the number of degrees of freedom is known. To solve this problem, a new manifold learning technique, named LSML, is introduced. LSML focuses on learning tangents rather than points themselves as done in conventional manifold learning.

Both methods are validated and compared with state of the art techniques on real and synthetic data.

Chapter 1

Introduction

In this dissertation, we address the problem of reconstructing the 3D structure of a scene from a movie taken with a single camera.

In computer vision, images can be analyzed at different levels of understanding. For example, in *segmentation* the image of a scene is divided into meaningful entities, like objects. Similarly, in *recognition*, an object is identified in an image, not necessarily by segmenting it. When analyzing videos, the temporal dimension brings a whole new set of information but also challenges.

The setup considered in this dissertation is the following: a unique monocular camera films a scene where an unknown object deforms over time. The goal is to recover, by using the *motion* of the camera filming the scene, the *3D-structure* of the deforming object. This is *Non-Rigid Structure From Motion*, or NRSFM.

The goal of this dissertation is to demonstrate how NRSFM can be considered as a *manifold learning* problem focused on recovering the low-dimensional shape embedding of an object. We will show how this new interpretation improves on current state of the art techniques and also overcomes the limitations of currently used assumptions.

Finally, while notations are conventional and will be explained along this dissertation, we refer the reader to Section A.1 for a list of used notations.

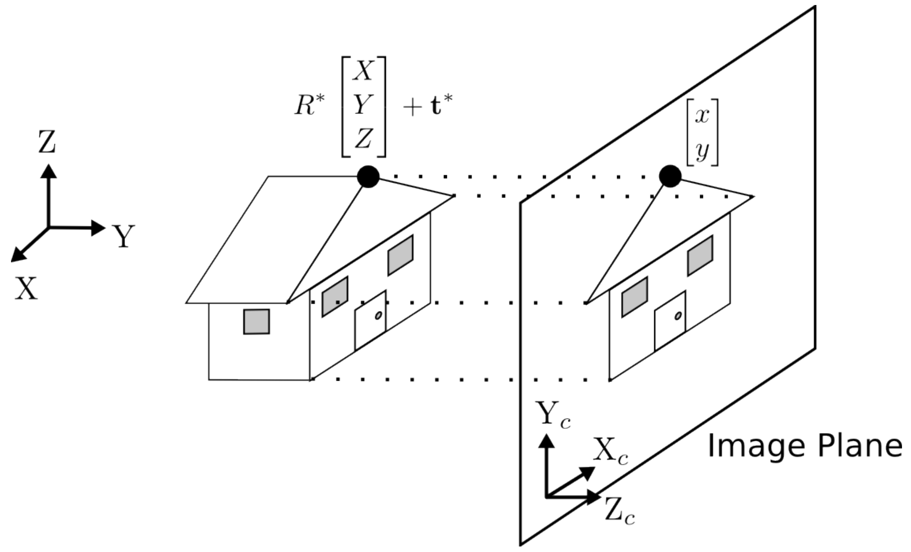

$R^* \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}^*$
$\begin{bmatrix} x \\ y \end{bmatrix}$
Z
Y
X
Y_c
X_c
Z_c
Image Plane

Figure 1.1: An orthographic camera is the simplest camera model: the scene is simply projected onto the image plane by removing the depth in the scene. There is therefore no perspective distortion. This model is usually used to simplify the *projective camera* which is used for conventional *pinhole cameras*, CCD cameras and X-ray images.

1.1 Camera Model and Image Formation

In *Structure From Motion* (*a.k.a.* SFM), the goal is to recover from a monocular video stream, the *structure* of a scene as well as the *motion* of the camera.

A camera projects an observed 3D point of coordinates $\begin{bmatrix} X & Y & Z \end{bmatrix}^\top$ onto the image plane where only its 2D coordinates $\begin{bmatrix} x' & y' \end{bmatrix}^\top$ are measured.

In this dissertation, we limit ourselves to the case of an orthographic camera. Figure 1.1 illustrates the properties of such a device while Figure 1.1 shows when the orthographic gets valid. The following equation defines the image formation for an orthographic camera:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \Pi K \left(R^* \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}^* \right), \text{ with } \Pi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (1.1)$$

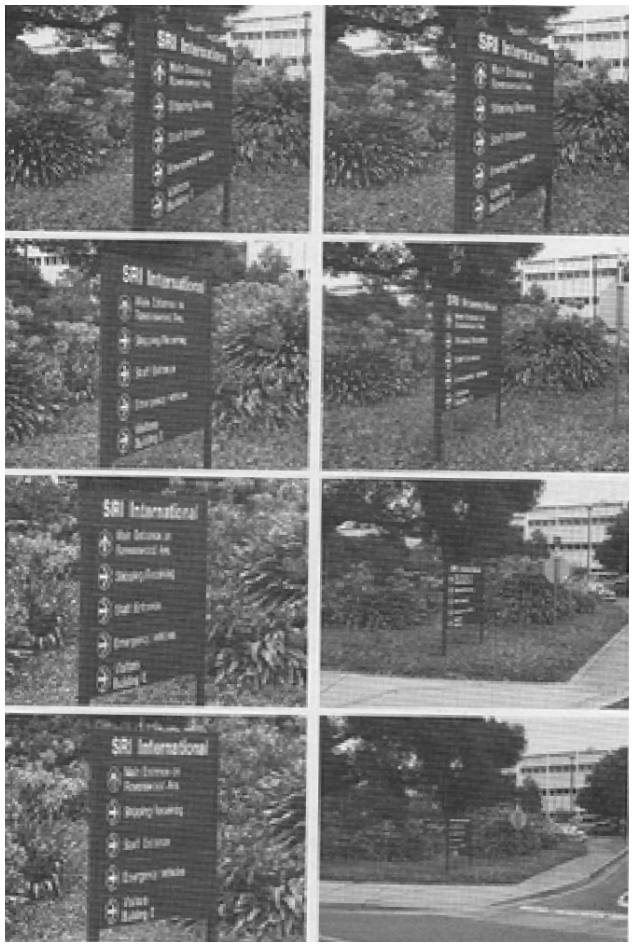


Figure 1.2: Suppose Z_{avg} is the average distance of an object from the camera and d_{avg} is the average width of the object (measured along the optical axis of the camera). As a rule of thumb, the orthographic camera model is reasonable to use when $Z_{avg} \geq 10d_{avg}$. In the figure above from Faugeras and Luong (2004), on the right, we see a scene containing a rectangular sign viewed by a projective camera as the photographer moves farther away from the sign. On the left, the portion of each image containing the sign is cropped and re-sized so that the sign occupies approximately the same area. It is evident as the camera moves farther away that the perspective effects diminishes. In particular, the vanishing point for the parallel lines on the top and bottom of the sign moves increasingly close to innity. Alfred Hitchcock pioneered the use of this effect (the so-called "Hitchcock zoom") in the movie Vertigo.

where $(R^*, \mathbf{t}^*)$ are a rotation and translation and define the position of the scene with respect to the camera, and K is the *internal parameter* matrix:

$$K = \begin{bmatrix} \alpha_x & s & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

where α_x and α_y are the scale factors for the x/y coordinate directions s is the skew.

In this dissertation, we also assume that the internal parameters are known in advance, *i.e.* the camera is *calibrated*. This simplifies the image formation equation to:

$$\begin{bmatrix} x \\ y \end{bmatrix} = \Pi \left(R^* \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t}^* \right) = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + \mathbf{t} \tag{1.3}$$

where $(R, \mathbf{t})$ are the first two rows of $(R^*, \mathbf{t}^*)$ (this notation will be used throughout the dissertation).

1.2 Feature Paradigm

While certain methods (Kutulakos and Seitz (2000); Cheung et al. (2005)) consider the scene data as dense, we use the feature-paradigm in this dissertation: only a few features are tracked during the video sequence and their image coordinates are considered as input of the SFM problem. Noise might be present.

Formally, the video sequence contains n features tracked over f frames. The $\begin{bmatrix} x & y \end{bmatrix}^\top$ coordinates of each feature in each image are horizontally stacked in a $2 \times n$ matrix called the *measurement matrix* W_t (t indexes time). The goal of the SFM problem is to recover the 3D positions of all the features over time (horizontally stacked in the $3 \times n$ *shape matrix* S_t) as well as the camera position parameters $(R_t, \mathbf{t}_t)$.

Equation (1.3) can now be extended to all the features in the object:

$$W_t = R_t S_t + \mathbf{t}_t \mathbf{1}_n^\top \tag{1.4}$$

where $\mathbf{1}_n$ is the n-dimensional vector composed of 1's. Per frame, $2 \times n$ measurements are given while the following parameters need to be recovered:

- 3 for rotation
- 2 for translation
- $3 \times n$ for the shape of the object

As $5 + 3n > 2n$, the problem is very underconstrained: to solve for the ambiguity, one needs additional constraints on the object like the following ones:

- the object is *rigid*. In this case, the shape matrix is constant and $S_t = S$. This is *Rigid SFM*.
- the object is *non-rigid* but its shape is assumed to be a linear combination of *basis shapes*. This is *classical non-rigid SFM*.
- the object is *non-rigid* and its shape is assumed to belong to a low-dimensional manifold, not necessarily linear as in the previous constraint.

This dissertation extends the current state of the art for the second constraint and introduces the third constraint as well as a solution for it.

Finally, it is worth noting that every solution $(R_t^*, \mathbf{t}_t^*, S_t)$ of Equation (1.4) is ambiguous up to any global rigid transformation $(R^*, \mathbf{t}^*)$ as $(R_t^* {R^*}^{-1}, \mathbf{t}_t - R_t^* {R^*}^{-1} \mathbf{t}, R^* S_t + \mathbf{t})$ would also be a solution. This is the *global ambiguity*.

1.3 Rigid SFM

In rigid SFM, the shape matrix to recover is constant: $S_t = S$. Therefore, we can write the following identities:

$$W_t = R_t S_t + \mathbf{t}_t \mathbf{1}_n^\top \tag{1.5}$$

$$\begin{bmatrix} W_1 \\ \vdots \\ W_f \end{bmatrix} = \begin{bmatrix} R_1 & \mathbf{t}_1 \\ \vdots & \\ R_f & \mathbf{t}_f \end{bmatrix} \begin{bmatrix} S \\ {\mathbf{1}_n}^\top \end{bmatrix} \tag{1.6}$$

As there is a *global ambiguity*, we can constrain the 3D points of the shape to be centered (the average of S along the second dimension is 0). If the measurements W_t are now normalized to $\overline{W_t}$ (the average of W_t along the second dimension is 0), the optimal $\mathbf{t}_t$'s simply are $\mathbf{t}_t = \mathbf{0}$ and the rigid SFM equation becomes:

$$\overline{W} \triangleq \begin{bmatrix} \overline{W_1} \\ \vdots \\ \overline{W_f} \end{bmatrix} = \underbrace{\begin{bmatrix} R_1 \\ \vdots \\ R_f \end{bmatrix}}_{Motion} \underbrace{S}_{Structure} \tag{1.7}$$

This equation is due to Tomasi and Kanade (1992) and illustrates how the measurement matrix has rank 3 and can be decomposed into the motion matrix and the structure matrix. Therefore, computing these two matrices can be reduced to a rank approximation of the measurement matrix by SVD. Nonetheless, the motion matrix needs to be composed of rotations, therefore, some orthonormality constraints also need to be imposed when decomposing the measurement matrix (as detailed in Tomasi and Kanade (1992)).

The *global ambiguity* is resolved by imposing that $R_1^* = \mathrm{I}$ leading to a unique ambiguity for $f \geq 3$:

$$\begin{bmatrix} \overline{W_1} \\ \vdots \\ \overline{W_f} \end{bmatrix} = \begin{bmatrix} R_1 \\ \vdots \\ R_f \end{bmatrix} Q Q^{-1} S, \text{ where } Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \tag{1.8}$$

which is called the *Necker reversal* and can simply be interpreted as the ambiguity of wether the object is in front or behind the camera.

To perfect the solution, what is called the *reprojection error* is minimized:

$$\text{err}_{reproj} = \left\| \overline{W} - \begin{bmatrix} \mathrm{I}_{2\times 3} \\ R_2 \\ \vdots \\ R_f \end{bmatrix} S \right\|_2^2 \tag{1.9}$$

using gradient descent over all the rotation parameters and the 3D shape. This method is called *bundle adjustment* (Triggs et al. (2000)) and already has a few years of development. Some techniques, like Lourakis and Argyros (2004), are both fast, accurate and memory efficient.

While rigid SFM is pretty much solved in its basic form for an orthographic camera, it has been extended to the projective camera model (Soatto and Perona (1994)), multiple rigid bodies (Costeira and Kanade (1998)) and articulated bodies (Yan and Pollefeys (2005)).

Its practicality has also been increased by embedding occlusion-handling (Chen and Suter (2004); Jacobs (1997)) and outlier robustness (using techniques like Fischler and Bolles (1981) or Li (2007)).

A lot of exciting recent new work has been put into making it more robust for the projective model by using the L_∞ norm (Kahl (2005)), more practical, and also to work with large scale datasets (Agarwal et al. (2008)).

1.4 Non-Rigid SFM

When assuming the object of study is non-rigid, the current state of the art Torresani et al. (2008) assumes that the 3D shape belongs to a linear subspace of dimensionality s and can therefore be explained as a linear combination of s canonical shapes $\mathtt{S}^i$ ($3 \times n$ matrices) added to a constant shape $\mathtt{S}^0$. The measurements can then be explained with the following image formation equation:

$$W_t = R_t \left(\mathtt{S}^0 + \sum_{i=1}^{s} l_t^i \mathtt{S}^i \right) + \mathbf{t}_t \mathbf{1}_n{}^\top \tag{1.10}$$

We choose the notation: $\mathbf{l}_t = \begin{bmatrix} l_t^1 & \dots & l_t^s \end{bmatrix}^\top$. It is worth noting that in earlier versions of NRSFM, like in Bregler et al. (2000); Brand (2001); Torresani et al. (2001); Xiao et al. (2006a), a shape S^t was considered as a linear combination of s shapes, which is simpler but less intuitive (and not just deformations as in the model we choose). This previous model was simply different because the first component of $\mathbf{l}_t$ was not necessarily 1.

As in the rigid case (Tomasi and Kanade (1992)), we can eliminate the $\mathtt{t}_t$ by subtracting the mean of the measurements. By imposing the $\mathtt{S}^i$ to also be centered, the measurement matrix can be factorized into the motion matrix and the shape basis matrix:

$$\overline{W} = \begin{bmatrix} \overline{W_1} \\ \vdots \\ \overline{W_f} \end{bmatrix} = \begin{bmatrix} R_1 & l_1^1 R_1 & \cdots & l_1^s R_1 \\ & & \vdots & \\ R_f & l_f^1 R_f & \cdots & l_f^s R_f \end{bmatrix} \begin{bmatrix} \mathtt{S}^0 \\ \vdots \\ \mathtt{S}^s \end{bmatrix} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 & \mathbf{l}_1^\top \end{bmatrix} \otimes R_1 \\ \vdots \\ \begin{bmatrix} 1 & \mathbf{l}_f^\top \end{bmatrix} \otimes R_f \end{bmatrix}}_{\text{Motion}} \underbrace{\begin{bmatrix} \mathtt{S}^0 \\ \vdots \\ \mathtt{S}^s \end{bmatrix}}_{\text{Shape Basis}} \qquad (1.11)$$

where $\otimes$ is the Kronecker product.

This formulation inspired the pioneers of NRSFM to treat the problem as a matrix factorization problem, just like for rigid SFM. Bregler et al. (2000) choose to factorize the matrix by first assuming the object to be rigid (like in Kim and Hong (2005)) or having a dominant rigid component. This provides an initialization to several optimization techniques like gradient descent Bregler et al. (2000) or an EM-type algorithm (Torresani et al. (2003)). These techniques usually have two problems: they require a good initialization (which might not be provided if the object does not comply to their assumptions) and their optimization usually does not consider all the parameters at once.

Nonetheless, they have been improved by combining a feature tracker Torresani et al. (2001); Del Bue et al. (2007) and including noise models Torresani et al. (2008). Finally, while Xiao et al. (2006a) and Brand (2005) provided a closed-form solution in the noiseless case, the same formulation was used to extend the analysis to the projective case (Xiao and Kanade, 2005; Llado et al., 2005; Vidal and Abretske, 2006).

1.5 Current Limitations

1.5.1 Problematic Closed Form

Wrong Rank Approximation in Presence of Noise

Many previous NRSFM methods tried to solve for an ambiguity matrix G in Equation (1.11):

$$\overline{W} = \begin{bmatrix} \overline{W_1} \\ \vdots \\ \overline{W_f} \end{bmatrix} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 & \mathbf{l}_1^\top \end{bmatrix} \otimes R_1 \\ \vdots \\ \begin{bmatrix} 1 & \mathbf{l}_f^\top \end{bmatrix} \otimes R_f \end{bmatrix}}_{\mathtt{M}=\text{ motion}} \underbrace{\begin{bmatrix} \mathtt{S}^0 \\ \vdots \\ \mathtt{S}^s \end{bmatrix}}_{\mathtt{S}=\text{ shape}} \triangleq \mathtt{MS} = AGG^{-1}B \tag{1.12}$$

where A and B have the right dimensions and can be obtained by SVD on W. While the above stands for a noise-free setup, there is no reason to believe that $\mathtt{M}$ and $\mathtt{S}$ are close to the best rank-4 approximation of W due to the special form of $\mathtt{M}$.

Wrong Rank Considerations

Moreover, the above is true for a full rank matrix $\mathtt{S}$. When the shape basis has some degeneracies (when the rank of $\mathtt{S}$, and therefore W, is not $3s$ but $r_\mathtt{S} \triangleq \text{rank}\,(\mathtt{S}) < 3s$ as explained in Xiao and Kanade (2004)), $\mathtt{S}$ is assumed to be formed of K_3 shapes of rank 3, K_2 of rank 2 and K_1 of rank 1. Where the mentioned paper is wrong is that it assumes that $r_\mathtt{S} = 3K_3 + 2K_2 + K_1$. It seems that the paper made a confusion by misinterpreting the rank of $\mathtt{S}$ (that considers the rows of $\mathtt{S}$) and the rank of the basis $\mathtt{S}$ (which considers the rows of $\mathtt{S}$ three by three).

While this seems tempting and its implementation showed some qualities, it can happen that each element of the basis is full rank but not $\mathtt{S}$ (as is the case with the bending shark from Torresani et al. (2003)).

Numerical Instability

We have implemented Xiao and Kanade (2004) and actually improved on the method (by using more modern math like SDP programming) but realized that this method is numerically unstable (due to the fact that only the ambiguity matrix is recovered, which is small compared to the dimensions of the problem) and performs poorly in certain degenerate cases (but perfectly in non-degenerate cases and some degenerate ones).

1.5.2 Problematic Iterative Solution

As mentioned above, the iterative solutions to this problem suffer from unstable initialization. The common assumption is to consider that the observed object has a main rigid component. While this is true for many objects (*e.g.* faces, persons ...), there is no reason to believe that this applies for all objects.

Iterative solutions also suffer from poor optimization: parameters are usually optimized alternatively (rotations, then shapes, then shape coefficients) which can result in local minima, slow convergence and “ping-pong” effects during optimizations.

1.5.3 Limited Model

Finally, and that will be a major point of this dissertation, while the linear model assumption appears to be valid for many real-world objects, there is no reason to believe that it is general enough.

We will demonstrate that allowing the shape manifold to be of a low dimensionality, but non-linear, can be solvable and applicable where current state-of-the-art fails.

Chapter 2

Linear Embedding

We first propose to solve the NRSFM problem by conserving the linear assumption that has been commonly used in previous works.

Let us take a step back from the matrix formulation of the problem. The linear assumption can be interpreted as having all the 3D-shapes the object can undergo as belonging to a linear embedding. While camera positions and the full S_t seem complex to compute, recovering the low s-dimensional shape embedding seems much simpler.

It is nonetheless a complex task and this chapter is dedicated to recovering the shape embedding from a video sequence. Mathematically, we will show how to recover the $\mathbf{l}_t$ coefficient independently from any other parameters. To that extent, we will exploit motion repetitions for SFM as it was used for action recognition Laptev and Perez (2007), motion segmentation Laptev et al. (2005) and sequence alignment Carceroni et al. (2004).

Additionally, we demonstrate how we can recover the rest of the parameters (shape basis $\mathtt{S}$ and camera rotations R_t) from this embedding, and therefore how we can perform full NRSFM.

Comparisons to state of the art methods will be presented as well as several experiments testing the robustness of this approach.

2.1 Problem Formulation

First we assume that a non-rigid object deforms in 3D shapes that can be observed several times in a video sequence.

As mentioned in Equation (1.11), the measured data can be explained with the following factorization:

$$\overline{W} = \begin{bmatrix} \overline{W_1} \\ \vdots \\ \overline{W_f} \end{bmatrix} = \begin{bmatrix} R_1 & l_1^1 R_1 & \cdots & l_1^s R_1 \\ & & \vdots & \\ R_f & l_f^1 R_f & \cdots & l_f^s R_f \end{bmatrix} \begin{bmatrix} \mathtt{S}^0 \\ \vdots \\ \mathtt{S}^s \end{bmatrix} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 & \mathbf{l}_1^\top \end{bmatrix} \otimes R_1 \\ \vdots \\ \begin{bmatrix} 1 & \mathbf{l}_f^\top \end{bmatrix} \otimes R_f \end{bmatrix}}_{\mathtt{M}= \text{ motion}} \underbrace{\begin{bmatrix} \mathtt{S}^0 \\ \vdots \\ \mathtt{S}^s \end{bmatrix}}_{\mathtt{S}= \text{ shape}} = \mathtt{MS} \tag{2.1}$$

As mentioned in Equation (1.8), the rigid structure of a scene can be reconstructed with only one ambiguity from three views or more.

Intuitively, if the 3D reconstruction from a triplet of frames has a low reprojection error (*cf*.Equation (1.9)), the three frames should represent a similar 3D shape. Similarly a high reconstruction error would probably be due to a poor matching of the three views. Due to the impossibility of relating this reconstruction error to a metric, we propose to only get an ordering between two triplets of frames; this is used next into a Multi-Dimensional Scaling (MDS) framework.

2.2 Outline

In traditional linear NRSFM methods, the basis elements $\mathtt{S}^i$ are only assumed to be centered at the origin. The Gram-Schmidt process can orthonormalize any basis but it also preserves the centering. Therefore, if a basis exists, a centered orthonormal basis can be built from it: the $\mathtt{S}^i$'s can consequently be assumed to be centered and orthonormal. But, as we use 1 for the first coordinate, we cannot impose $\|\mathtt{S}^0\|_F = 1$. Therefore, we can only impose the $\mathtt{S}^i$'s to be orthogonal, and $\|\mathtt{S}^i\|_F = 1, \forall i \neq 0$

Now, in this basis, every shape S_t has coordinates $[1, \mathbf{l}_t^\top]$ and therefore:

$$\|S_i - S_j\|_F = \|\mathbf{l}_i - \mathbf{l}_j\|_2 \tag{2.2}$$

Let us consider the function:

$$a_F(i,j,k) = \sum_{h \in \{i,j,k\}} \left\| S_h - \frac{S_i + S_j + S_k}{3} \right\|_F^2 \tag{2.3}$$

Let us assume for now that we can build a set of pairs of triplets of frames as follow:

$$\mathcal{F} = \{((i,j,k), (i',j',k')) \,|\, a_F(i,j,k) \leq a_F(i',j',k')\} \tag{2.4}$$

Basically, we are considering ordering between triplets of frames according to the function a_F. As we will see in Section 2.3, this set does not contain every pair $\{(i,j,k),(i',j',k')\}$ and it might even contain some outliers. Therefore, only certain triplets of frames have to be considered; this makes Generalized Non-metric Multi-Dimensional Scaling (GNMDS) from Agarwal et al. (2007) the appropriate choice to solve for the $\mathbf{l}_t$'s.

2.2.1 Generalized Non-metric Multi-Dimensional Scaling Overview

Let us consider the positive semi-definite Gram matrix $K = [K_{ij}]_{1 \leq i,j \leq f} = [\mathbf{l}_i^\top \mathbf{l}_j]_{1 \leq i,j \leq f}$ and let use define:

$$a_2(i,j,k) = \sum_{h \in \{i,j,k\}} \left\| \mathbf{l}_h - \frac{\mathbf{l}_i + \mathbf{l}_j + \mathbf{l}_k}{3} \right\|_2^2 \tag{2.5}$$

By using Equation (2.2), $a_F(i,j,k) = a_2(i,j,k)$.

Solving for the $\mathbf{l}_t$'s can therefore be reduced to finding a positive semi-definite matrix K such that:

$$a_2(i,j,k) \leq a_2(i',j',k') \text{ if } ((i,j,k),(i',j',k')) \in \mathcal{F} \tag{2.6}$$

By introducing slack variables, solving for the $\mathbf{l}_t$'s is equivalent to solving the following Semi-Definite Programming (SDP) problem:

$$\begin{aligned} \min_{K,\xi_{(i,j,k),(i',j',k')}} \quad & \sum_{((i,j,k),(i',j',k'))\in\mathcal{F}} \xi_{(i,j,k),(i',j',k')} \\ \text{subject to} \quad & a_2\left(i,j,k\right) \leq a_2\left(i',j',k'\right) + \xi_{(i,j,k),(i',j',k')} \\ & \xi_{(i,j,k),(i',j',k')} \geq 0, \quad K \succeq 0 \end{aligned} \tag{2.7}$$

Also, the constraints are linear in the elements of K as:

$$\begin{aligned} a_2\left(i,j,k\right) = & \left(\frac{4K_{ii}+K_{jj}+K_{kk}-4K_{ij}-4K_{ik}+2K_{jk}}{9}\right) + \\ & \left(\frac{K_{ii}+4K_{jj}+K_{kk}-4K_{ij}+2K_{ik}-4K_{jk}}{9}\right) + \\ & \left(\frac{K_{ii}+K_{jj}+4K_{kk}+2K_{ij}-4K_{ik}-4K_{jk}}{9}\right) + \\ = & \frac{2}{3}\left(K_{ii}+K_{jj}+K_{kk}-K_{ij}-K_{ik}-K_{jk}\right) \end{aligned} \tag{2.8}$$

We can also notice for the future that this is equivalent to:

$$a_F\left(i,j,k\right) = \frac{1}{3}\left(\|S_i - S_j\|_F^2 + \|S_i - S_k\|_F^2 + \|S_j - S_k\|_F^2\right) \tag{2.9}$$

2.2.2 Constraints

Note the K_{ij}'s can be recovered up to a similarity. We choose to constrain the rotation/translation/scale differently from Agarwal et al. (2007):

- **Scale**. If $(\mathbf{l}_t, R_t, \mathtt{S})$ is a solution of Equation (2.1), so is $(\alpha\mathbf{l}_t, R_t, \frac{1}{\alpha}\mathtt{S}), \forall\alpha \neq 0$. Therefore, scale ambiguity is already inherent to our formulation. Nonetheless, to prevent the $\mathbf{l}_t$'s from collapsing to the origin, and as we have chosen the $\mathtt{S}^i$'s to be of norm 1, we choose to enforce the scale constraint as follows:

$$\|\mathbf{l}_i - \mathbf{l}_j\|_2^2 = \|S_i - S_j\|_2^2 \geq a_{\min}(i,j) \tag{2.10}$$

  where $a_{\min}(i,j)$ is defined in Section 2.3.1.

- **Rotation**. The rotation ambiguity is also inherent to our formulation as the shape formation equation contains an invertible matrix ambiguity:

$$\text{vec}\,(S_t) = \begin{bmatrix}\text{vec}\,(\mathtt{S}^0)\cdots\text{vec}\,(\mathtt{S}^s)\end{bmatrix}\begin{bmatrix}1\\ \mathbf{l}_t\end{bmatrix} = \begin{bmatrix}\text{vec}\,(\mathtt{S}^0)\cdots\text{vec}\,(\mathtt{S}^s)\end{bmatrix} Q^{-1}Q\begin{bmatrix}1\\ \mathbf{l}_t\end{bmatrix} \tag{2.11}$$

where vec $(\cdot)$ is the operator that stacks the columns into a vector. No constraints therefore needs to be enforced to resolve this ambiguity.

- **Translation**. Finally, the translation ambiguity is also inherent to our formulation, and therefore unimportant, as for every vector $\mathbf{t}$:

$$\text{vec}\,(S_t) = \begin{bmatrix}\text{vec}\,(\mathtt{S}^0)\cdots\text{vec}\,(\mathtt{S}^s)\end{bmatrix}\begin{bmatrix}1\\ \mathbf{l}_t+\mathbf{t}\end{bmatrix} = \begin{bmatrix}\text{vec}\left(\mathtt{S}^{0\prime}\right)\cdots\text{vec}\,(\mathtt{S}^s)\end{bmatrix}\begin{bmatrix}1\\ \mathbf{l}_t\end{bmatrix} \tag{2.12}$$

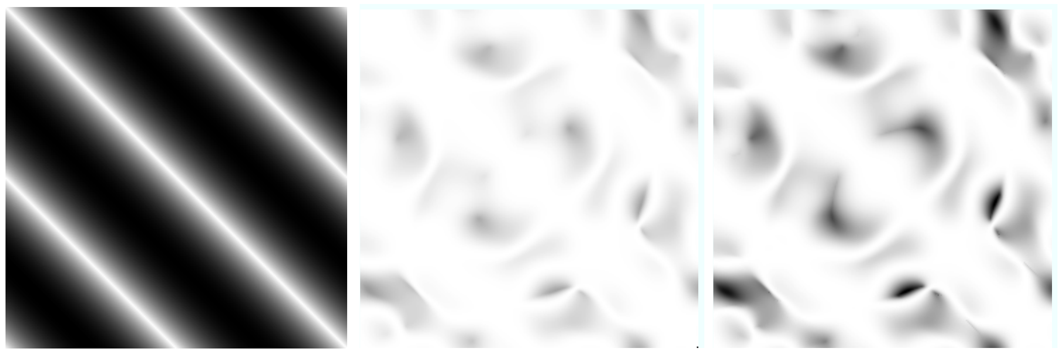
where vec $\left(\mathtt{S}^{0\prime}\right) = \text{vec}\,(\mathtt{S}^0) + \begin{bmatrix}\text{vec}\,(\mathtt{S}^1)\cdots\text{vec}\,(\mathtt{S}^s)\end{bmatrix}\mathbf{t}$. Nonetheless, to avoid a drifting of the $\mathbf{l}_t$'s during their computation, we impose that the $\mathbf{l}_t$'s be centered:

$$\sum_{t=1}^{f}\mathbf{l}_t = 0 \Longleftrightarrow \left(\sum_{t=1}^{f}\mathbf{l}_t\right)^{\top}\left(\sum_{t=1}^{f}\mathbf{l}_t\right) = 0 \Longleftrightarrow \sum_{t,t'}K_{tt'} = 0 \tag{2.13}$$

We also add a regularization term to impose a smooth deformation of the object over time: $\lambda\sum_{t=2}^{f-1}\Delta''(\mathbf{l}_t)$, where λ weighs the regularization and Δ'' is a finite difference approximation of the second derivative of $\mathbf{l}_t$ at t (and can be expressed in terms of $K_{tt'}$). We therefore have the new SDP formulation:

$$\begin{aligned}\min_{K,\xi_{(i,j,k),(i',j',k')}} \quad & \sum_{((i,j,k),(i',j',k'))\in\mathcal{F}} \xi_{(i,j,k),(i',j',k')} + \lambda\sum_{t=2}^{f-1}\Delta''(\mathbf{l}_t)\\ \text{subject to} \quad & a_2\,(i,j,k) \le a_2\,(i',j',k') + \xi_{(i,j,k),(i',j',k')}\\ & \|\mathbf{l}_i - \mathbf{l}_j\|_2^2 \ge a_{\min}(i,j), \quad \forall(i,j)\\ & \sum_{t,t'}K_{tt'} = 0, \quad \xi_{(i,j,k),(i',j',k')} \ge 0, \quad K \succeq 0\end{aligned} \tag{2.14}$$

The $\mathbf{l}_t$'s are then obtained from K by using its $r_{\mathtt{S}}$-rank SVD decomposition (where $r_{\mathtt{S}} = \text{rank}\,(\mathtt{S}) = \text{rank}\,(W)$).



(a) Ideal Pairwise Affinity (b) Uncalibrated 2-View Pairwise Affinity (c) Calibrated 2-View Pairwise Affinity

Figure 2.1: **Pairwise Affinities.** These figures relate to the 240-frame shark sequence from Torresani et al. (2003) where a shark rotates its tail circularly twice (as illustrated in Figure 2.3). As there is an exact repetition in the observed shape over time, one would expect frames separated by one period to have a high affinity (white in Figure (a)) while others should have a low affinity (black). For example, the first frame (first row of Figure (a)) should match best frames 1 and frame 121, frame 2 should match best frame 2 and 122 *etc*. If one chooses to use the uncalibrated reprojection error as an affinity, many false positives arise, due to ambiguities in orthographic reconstruction as shown in Figure 2.1(b). Finally, if one uses calibrated reconstruction, as shown in Figure (c), some ambiguities can be removed but it is still not fully usable.

2.3 Ordering Set $\mathcal{F}$

We have shown in the previous section how the coordinates of every shape S_t in the basis $\mathtt{S}$ can be computed by using an ordering between triplets of frames. In the projective case, one could simply compute the best reconstruction from two frames and compute the reprojection error to see how well it matches the measurements. A low error would indicate that the two frames are more likely to be of the same shape while a large error would indicate a higher dissimilarity.

In the orthographic case, as there is no depth information, the 3D reconstruction from two views has ambiguities (like Necker reversal or bas-relief ambiguity) as shown in Figure 2.1. This problem can, however, be alleviated using triplets of frames. In the calibrated orthographic case, three centered measurements W_i, W_j and W_k give rise to only two optimal reconstructions S_{ijk} and QS_{ijk} as:

$$\begin{bmatrix} W_i \\ W_j \\ W_k \end{bmatrix} = \begin{bmatrix} R_i \\ R_j \\ R_k \end{bmatrix} QQS_{ijk}, \text{ where } Q = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \pm 1 \end{bmatrix} \tag{2.15}$$

where $R_i = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and R_j, R_k are the top two rows of two rotation matrices.

Let us now define an infimum and a supremum to $a_F(i, j, k)$ based on measurements only. To this end, we need to define the 3D measurement matrix W_i^* whose first two rows are W_i, but whose third row is what would be obtained if the depth could be measured. If there is no noise, $W_i^* = R_i^* S_i$. For the sake of simplicity, we will assume that there is no noise, but what follows would hold by adding an extra term of lower order.

2.3.1 Triplet Distance Infimum

Considering two frames i and j, we have:

$$\|S_i - S_j\|_F^2 = \left\| R_i^{*\top} W_i^* - R_j^{*\top} W_j^* \right\|_F^2 = \left\| W_i^* - R_i^* R_j^{*\top} W_j^* \right\|_F^2 \tag{2.16}$$

Hence:

$$\|S_i - S_j\|_F^2 \geq \begin{array}{rl} \min\limits_{R^*, \mathbf{x}, \mathbf{y}} & \left\| \begin{bmatrix} W_i \\ \mathbf{x}^\top \end{bmatrix} - R^* \begin{bmatrix} W_j \\ \mathbf{y}^\top \end{bmatrix} \right\|_F^2 \\ \text{subject to} & R^* \text{ is a rotation matrix} \\ & \text{centering conditions: } \overline{\mathbf{x}} = 0, \overline{\mathbf{y}} = 0 \end{array} \triangleq a_{\min}(i, j) \tag{2.17}$$

The expression on the right can be interpreted as a pose estimation problem where the third coordinates/depths are unknown.

It is a Procrustes problem with missing entries and linear constraints. $\mathbf{x}$ and $\mathbf{y}$ can be computed in closed form with respect to R^*. The minimum can be computed by gradient descent on the quaternion of R^* with initialization provided by neighboring pairs of frames: $(i-1,j),(i+1,j),(i,j-1),(i,j+1)$. The very first pair is chosen at random and the process is stopped when all the minima for the possible pairs are stable.

We denote this minimum by $a_{\min}(i,j)$. Hence:

$$a_F\left(i,j,k\right) \geq \frac{1}{3}\left(a_{\min}(i,j) + a_{\min}(j,k) + a_{\min}(k,i)\right) \tag{2.18}$$

We now have an infimum for $a_F\left(i,j,k\right)$, that we name $a_{\min}(i,j,k)$.

2.3.2 Triplet Distance Supremum

For the supremum, we can notice that:

$$a_F\left(i,j,k\right) = \min_S \frac{1}{3}\left(\|S_i - S\|_F^2 + \|S_j - S\|_F^2 + \|S_k - S\|_F^2\right) \tag{2.19}$$

Therefore:

$$a_F\left(i,j,k\right) \leq \frac{1}{3}\left(\|S_i - S_{ijk}\|_F^2 + \|S_j - S_{ijk}\|_F^2 + \|S_k - S_{ijk}\|_F^2\right) \tag{2.20}$$

where S_{ijk} is the optimal reconstruction for the 3 frames i,j and k. We define R'_i, R'_j, R'_k are the corresponding optimal rotation matrices (different from R_i, R_j, R_k but close for shapes that are close-by).

Now, if we assume that the reconstruction error in depth is similar to the reprojection errors in abscissa and ordinate, each term $\|S_i - S_{ijk}\|_F$ can be approximated by:

$$\|S_i - S_{ijk}\|_F \simeq \|W_i^* - R'_i S_{ijk}\|_F \simeq \frac{3}{2}\left\|W_i - {R'_i}^* S_{ijk}\right\|_F \tag{2.21}$$

The terms on the right in the expression above can be computed as part of the 3D reconstruction of frames i,j,k and therefore, we obtain a supremum for $a_F\left(i,j,k\right)$:

$$a_F\left(i,j,k\right) \leq a_{\max}(i,j,k) \tag{2.22}$$

2.3.3 Computing the Ordering Set $\mathcal{F}$

Now, given two triplets of pairs (i, j, k) and (i', j', k'), we can compute:

$$\begin{aligned} a_{\min}(i, j, k) \leq a_F\,(i, j, k) \leq a_{\max}(i, j, k) \\ a_{\min}(i', j', k') \leq a_F\,(i', j', k') \leq a_{\max}(i', j', k') \end{aligned} \tag{2.23}$$

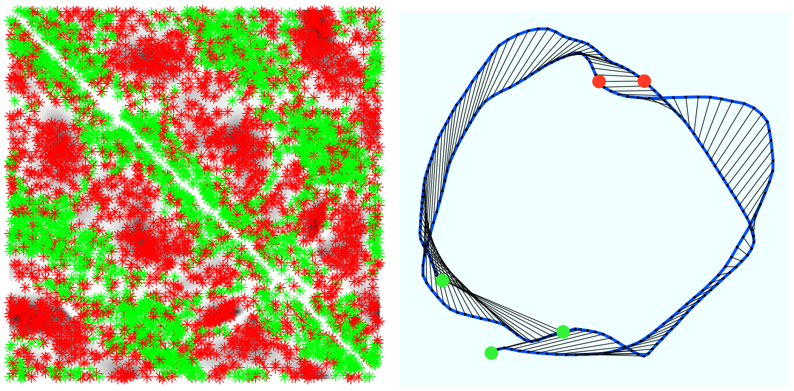
Therefore, if $a_{\max}(i, j, k) \leq a_{\min}(i', j', k')$, we have $((i, j, k), (i', j', k')) \in \mathcal{F}$ (and respectively with $a_{\max}(i', j', k') \leq a_{\min}(i, j, k)$).

Such triplets exist as $a_{\max}(i, j, k) = 0$ for pairs representing the same shape. Similarly, $a_{\min}(i, j, k) \gg 0$, usually, for frames representing very different shapes.

2.3.4 Practicality

In practice, the $a_{\min}(i, j, k)$ and $a_{\max}(i, j, k)$ are computed for several triplets and if an ordering exists, it is used in the SDP problem.

While we could compute $a_{\max}(i, j, k)$ for every triplet, the task is computationally very expensive (it is $O(f^3)$). Therefore, to speed up the process and obtain low $a_{\max}(i, j, k)$, we first compute all the $a_{\min}(i, j)$'s, and, given a frame i, we draw j and k from a distribution based on this error, leading to i, j and k more likely to be views of the same shape. Figure 2.2 illustrates this procedure.



(a) Sampling Procedure (b) Recovered $\mathbf{l}_t$'s

Figure 2.2: These two figures are also based on the shark sequence. **Frame Triplet Selection Procedure.** Figure (a) illustrates how the sampling is performed: for every frame i in the vertical axis, two frames j and k leading to a low reconstruction error are chosen: these two are plotted in green on the line number i. Then, two views j' and k' are chosen so that $a_{\max}(i, j, k) \leq a_{\min}(i, j', k')$: they are plotted in red on the same line. If no such pair (j', k') exists, the pair (j, k) is not considered. As we can see, the green couples are almost only present on the diagonals and sub-diagonals while the red are located anywhere. 2000 triplets are sampled and represented on this figure. This amount was chosen with respect to limitations of the SDP solver. **Recovered Embedding.** Figure (b) illustrates the recovered $\mathbf{l}_t$'s for $s = 2$, each linked to its previous and next temporal neighbors (in blue) as well as to the same point a period after (in black). What we see is one path almost overlapping with itself once (hence the periodicity of the motion). In green are displayed the points 1, 121 and 240, while in red are displayed the points 61 and 181 (these points are important as the video sequence has 240 frames and a period of 1200. The green points should indicate when the shark returns to its initial state (hence their proximity) while the red ones indicate the half period (hence the fact they are furthest from the green points and actually symmetric). While these points are not perfectly overlapping, they are close and enough for us to perform full SFM. More samples also improve this embedding as shown in Figure 2.8.

2.4 Optimal Shape Basis and Rotation

Let us re-mention Equation (2.1)

$$\overline{W} = \begin{bmatrix} \overline{W_1} \\ \vdots \\ \overline{W_f} \end{bmatrix} = \underbrace{\begin{bmatrix} \begin{bmatrix} 1 & \mathbf{l}_1^\top \end{bmatrix} \otimes R_1 \\ \vdots \\ \begin{bmatrix} 1 & \mathbf{l}_f^\top \end{bmatrix} \otimes R_f \end{bmatrix}}_{\mathtt{M}=\text{ motion}} \underbrace{\begin{bmatrix} \mathtt{S}^0 \\ \vdots \\ \mathtt{S}^s \end{bmatrix}}_{\mathtt{S}=\text{ shape}} = \mathtt{MS} \tag{2.24}$$

Now that we have obtained a good approximation of the $\mathbf{l}_t$'s, we are going to use them to first find a good estimate of the optimal shape basis $\mathtt{S}$ and then obtain the R_t's. We will then use these parameters as an initialization to a gradient descent procedure.

The proposed full structure from motion method now detailed does not suffer from the rank problems from Xiao et al. (2006a) and that were approached but not clearly solved in Xiao and Kanade (2004). It also does not face problems when som shape coefficients are small when trying to recover the rotation matrices (like in Xiao and Kanade (2004)).

In the following, we will assume that $\mathtt{S}$ is of rank $r_{\mathtt{S}}$ ($r_{\mathtt{S}} \leq 3(s+1)$; it was assumed that $r_{\mathtt{S}} = 3(s+1)$ in Xiao et al. (2006a)) and that it is given by the user or guessed from W (as $\text{rank}\,(\mathtt{S}) = \text{rank}\,(W)$)

2.4.1 Rank Constraint

Now, let us consider an optimal rank decomposition of $\overline{W}$ (*e.g.* provided by SVD): $\overline{W} = AB$ with A and B full rank (of rank $r_{\mathtt{S}}$). As the rows of B and $\mathtt{S}$ span the same spaces, we have $\mathtt{S} = GB$ where G is a $3(s+1) \times r_{\mathtt{S}}$ ambiguity matrices.

We will first recover G, which will give us $\mathtt{S}$ and therefore, knowing the $\mathbf{l}_t$, the S_t's. Recovering the R_t will then just be an instance of pose estimation.

2.4.2 Kronecker Constraint

By using the definition of $\mathtt{S}$ in Equation (1.12), $\mathtt{S}$ must verify for every t:

$$\left(\begin{bmatrix}1 & \mathbf{l}_t^\top\end{bmatrix} \otimes R_t\right) \mathtt{S} = \overline{W}_t \quad \text{or again} \quad R_t \left(\begin{bmatrix}1 & \mathbf{l}_t^\top\end{bmatrix} \otimes \mathrm{I}_3\right) G = \overline{W}_t B^+ \tag{2.25}$$

as B has full row rank. We also add a regularization term so that the rotation matrices do not change much from frame to the next, and obtain the following bilinear problem:

$$\min_{R_t, G} \sum_{t=1}^{f} \left\| R_t \left(\begin{bmatrix}1 & \mathbf{l}_t^\top\end{bmatrix} \otimes \mathrm{I}_3\right) G - \overline{W}_t B^+ \right\|_F^2 \tag{2.26}$$

We also add a regularization term so that the rotations do not change much from frame to frame: $\lambda_R \sum_{t=2}^{f} \|R_t - R_{t-1}\|_F^2$. In order not to have the R_t's shrink to 0 and G diverge to infinity (as for any (R_t, G), $(\alpha R_t, \frac{1}{\alpha} G)$ is also a solution), we also need a counter-balancing regularization term: $\lambda_G \|G\|_F^2$. In practice, we choose $\lambda_R = \lambda_G = 1$. Hence the new bilinear problem:

$$\min_{R_t, G} \sum_{t=1}^{f} \left\| R_t \left(\begin{bmatrix}1 & \mathbf{l}_t^\top\end{bmatrix} \otimes \mathrm{I}_3\right) G - \overline{W}_t B^+ \right\|_F^2 + \lambda_R \sum_{t=2}^{f} \|R_t - R_{t-1}\|_F^2 + \lambda_G \|G\|_F^2 \tag{2.27}$$

While recent work like Chandraker and Kriegman (2008) improves the solving of bilinear problems if one of the two sets of variables as a much lower dimensionality, it is impractical in our case as the dimensions of G are too high.

We therefore solve it by generating several random initializations for G and proceed by alternate optimization between the R_t's and G. We also drop, for now, the rotation constraint on the R_t's.

Knowing G and by posing $C = \left(\begin{bmatrix}1 & \mathbf{l}_t^\top\end{bmatrix} \otimes \mathrm{I}_3\right) G$, R_t is a solution of:

$$2\left(R_t C - \overline{W}_t B^+\right) C^\top + 2\lambda_R \left(R_t - R_{t-1} + R_t - R_{t+1}\right) = 0 \tag{2.28}$$

$$R_t \left(C C^\top + 2\lambda_R \mathrm{I}_3\right) - \lambda_R \left(R_{t-1} + R_{t+1}\right) = \overline{W}_t B^+ C^\top \tag{2.29}$$

Similarly, knowing the R_t's, G is a solution of:

$$2\mathtt{M}^\top(\mathtt{M}G - \overline{W} B^+) + 2\lambda_G G = 0 \tag{2.30}$$

$$\left(\mathtt{M}^\top \mathtt{M} + \lambda_G \mathrm{I}_{3(s+1)}\right) G = \mathtt{M}^\top \overline{W} B^+ \tag{2.31}$$

In practice, we use 10 random initializations and 50 alternate optimization iterations.

It is worth noting that after this optimization, an approximation to the best solution is obtained up to an ambiguity matrix Q as:

$$R_t \left(\begin{bmatrix}1 & \mathrm{l}_t^\top\end{bmatrix} \otimes \mathrm{I}_3\right) G = R_t Q \left(\begin{bmatrix}1 & \mathrm{l}_t^\top\end{bmatrix} \otimes \mathrm{I}_3\right) \left(\mathrm{I}_{s+1} \otimes Q^{-1}\right) G \tag{2.32}$$

2.4.3 Rotation Constraint

So far, we have recovered R_t's and G optimizing $\left(\begin{bmatrix}1 & \mathrm{l}_t^\top\end{bmatrix} \otimes R_t\right) \mathtt{S} = \overline{W}_t$ but the R_t's were not imposed to be rotation matrices.

We now seek the ambiguity matrix Q such that the R_t's are as close to rotation matrices as possible by optimizing:

$$\sum_{t=1}^{f} \left\| R_t Q Q^\top R_t^\top - \mathrm{I}_2 \right\|_F^2 \tag{2.33}$$

which is an SDP in $QQ^\top$.

2.4.4 Orthonormality of $\mathtt{S}$

Finally, the orthonormality on the basis can be interpreted as having the rows of $\begin{bmatrix}C^1 S & C^2 S & C^3 S\end{bmatrix} = \begin{bmatrix}C^1 GB & C^2 GB & C^3 GB\end{bmatrix} = \begin{bmatrix}C^1 G & C^2 G & C^3 G\end{bmatrix} (\mathrm{I}_3 \otimes B)$ be orthonormal. In the previous expression, $C^1 = \mathrm{I}_s \otimes \begin{bmatrix}1 & 0 & 0\end{bmatrix}$, $C^2 = \mathrm{I}_s \otimes \begin{bmatrix}0 & 1 & 0\end{bmatrix}$, $C^3 = \mathrm{I}_s \otimes \begin{bmatrix}0 & 0 & 1\end{bmatrix}$.

We therefore have the property:

$$\begin{bmatrix}C^1 G & C^2 G & C^3 G\end{bmatrix} (\mathrm{I}_3 \otimes B)(\mathrm{I}_3 \otimes B)^\top \begin{bmatrix}C^1 G & C^2 G & C^3 G\end{bmatrix}^\top = D \tag{2.34}$$

where D is I_{s+1} except for the element $(1,1)$ which is unknown (as $\|\mathtt{S}^0\|_F \neq 1$). This results in a quadratic SDP constraint in $\begin{bmatrix}C^1 G & C^2 G & C^3 G\end{bmatrix}$ which we actually do not use in the recovery of G.

2.4.5 Final Optimization

G is first recovered using the Kronecker and the rotation constraints. From there, $\mathtt{S}$ is recovered and therefore all the S_t. Recovering an initial estimate of R_t's is then multiple instance of pose estimation. While we could perform the full computation for every frame, we first only perform it for a few frames, but very accurately (we had little chance with EPnP from Lepetit et al. (2008) so we used our own implementation relying on a simple polynomial formulation and a solving by Henrion and Lasserre (2003)). We then compute the optimal R_t's by performing gradient descent on $||R_tS - W_t||_F^2$ with an initial estimate of R_{t-1} and R_{t+1}.

Once an initial estimate of all the parameters $\mathbf{l}_t, R_t$ and $\mathtt{S}$ is obtained, what follows is a gradient descent over all the parameters at once to minimize the factorization error $||W - MS||_F^2$. To take advantage of the sparsity of the problem, we interpret it as a sparse bundle-adjustment with points of dimensionality $3(s + 1)$ instead of 3 in the normal 3D-case (the camera parameters being extended to $(R_t, \mathbf{t}_t, \mathbf{l}_t)$) and then use Lourakis and Argyros (2004) to obtain a fast and accurate solution.

2.5 Experiments

We tested our approach on two synthetic datasets: the classical *Shark Data* from Torresani et al. (2003) and on our own data named *Roller Coaster*. We also experimented with real data.

The reconstruction error considered in these experiments is computed in percentage points: the average distance of the reconstructed point to the correct point divided by the span of the shape, as defined in Torresani et al. (2001). We also compare our approach (named TSFM for Triplet-based Structure From Motion) to two standard algorithms: Torresani et al. (2008) and Xiao and Kanade (2004). For the latter, we have our own implementation of the code which uses more modern math and does reach an error of 0 (as it is a closed-form solution) in the assumption of the paper (that were previously explained and that do not match the shark data).

The running time for these experiments is usually a few minutes on a 3GHz machine except for the following parts of the algorithm:

- the computation of the $a_{\min}$ takes 10 minutes. It is very optimized and results in an average of 3 steps of gradient descent for every pair. The speed could be improved with a specifically tuned gradient descent algorithm and a conversion to a lower-level language (like C++).
- the final gradient descent takes 15 minutes. The efficient Sparse Bundle Adjustment code from Lourakis and Argyros (2004) is used. The computation time seems hard to improve but it is worth oticing that 50 iterations are used while much fewer could be used (*cf*.Figure 2.7.

2.5.1 Shark Data

The shark data was first introduced in Torresani et al. (2003). It features a shark rotating its tail twice over 240 frames. In terms of performance, our method reaches 0.00% of error while Torresani et al. (2008)'s with 100 EM iterations reaches 1.67%

(1.24% in their paper) . Both cases were computed for $s = 2$. Several reconstructions and the errors are detailed in Figure 2.3.

2.5.2 Roller-Coaster Data

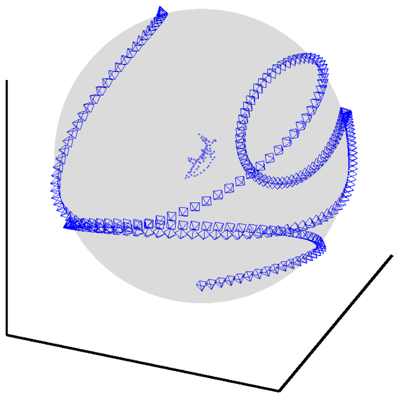
The next synthetic data can be thought of as a bike chain bent in its middle (*cf*.Figure 2.4(a)). The points are forced to evolve on a one-dimensional track and they perform 6 and their speed varies randomly. As shown in Figure 2.4(b), PCA cannot explain the data very well: this proves that the shape manifold is non-linear and that the shape basis assumption is not well suited for this data. However, we think it is interesting that our method still provides a good approximation of the shape manifold as shown in Figure 2.4(c) where the periodicity of the motion appears.

As far as 3D reconstruction is concerned, since this scenario does not fit the assumptions of neither our approach or Torresani et al. (2008), neither performs better than a 15 % error. Torresani et al. (2008) performs slightly better for a certain shape basis size, probably because it encompasses a temporal smoothness assumption. This assumption is only considered in our case when computing the $\mathbf{l}_t$'s but ignored after. A regularization term in the final optimization is a further improvement that could be implemented.

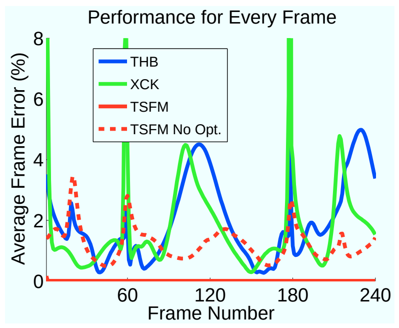
2.5.3 Real Data

The final experiment is a video of a sheet of paper twisted laterally back and forth. 51 features are tracked during the 190 frames of the sequence. The motion is repetitive but not circular: the sheet is bent to an extreme before being bent to normal. Figure 2.5 illustrates how this impacts the reconstruction of the shape embedding.

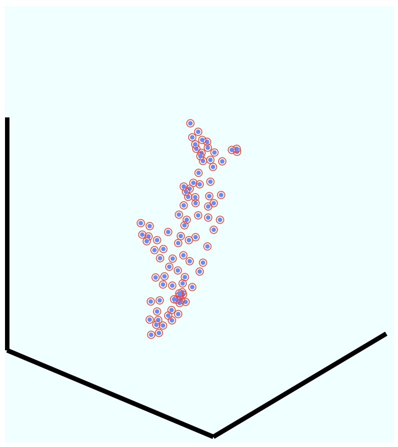
This deformation seems fairly easy to explain with the shape basis and choosing $s = 2$ was enough to reproduce equivalent 3D deformations.



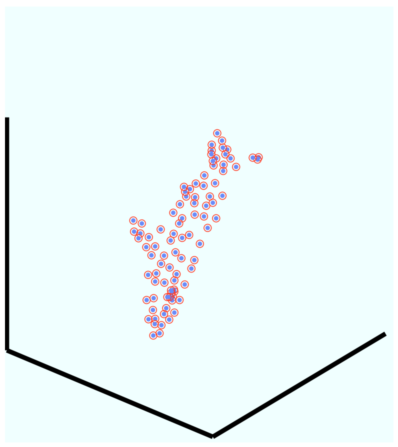
(a) Camera Positions



(b) Depth Error over Time

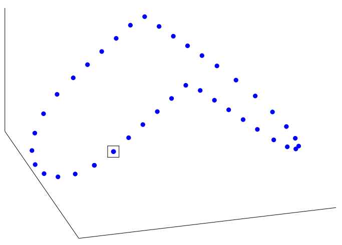


(c) Frame 100

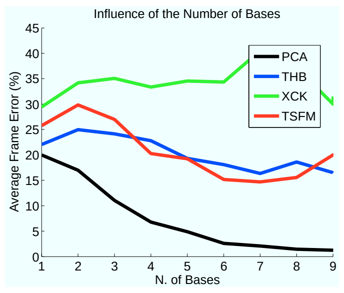


(d) Frame 200

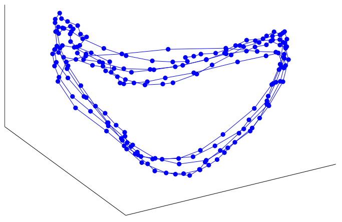
Figure 2.3: **Shark Dataset from from Torresani et al. (2003).** Figure (a) shows the camera positions used in our experiment: the camera has a smooth random path over a sphere. Figure (b) shows the reconstruction error produced by Xiao-Chai-Kanade CVPR04 (XCK), Torresani-Hertzmann-Bregler PAMI08 (THB) and our algorithm (TSFM), with and without the final gradient descent. Figure (c),(d) illustrate some novel views (in blue) overimposed with the ground truth in red of the shark data.



(a) Example of the Roller-Coaster Sequence

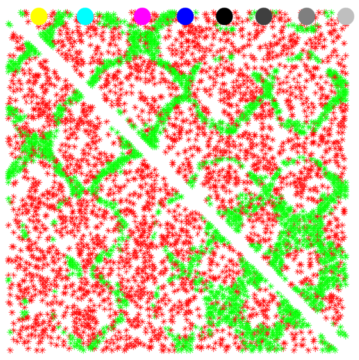


(b) Reconstruction Error for Different Techniques

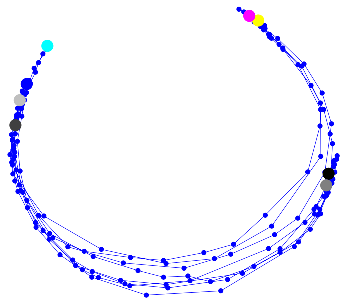


(c) Shape Embedding Based on the First 3 Coordinates

Figure 2.4: **Roller Coaster Dataset**. Figure 2.4(a) is an example of the video sequence: points are moving on the same track like a roller coaster (the square is simply used to help visualize the movement of the points on the “rail”). Figure 2.4(b) compares the errors provided with PCA, Xiao-Chai-Kanade CVPR04 (XCK), Torresani-Hertzmann-Bregler PAMI08 (THB) and our algorithm (TSFM). While our approach performs as poorly as the other ones as shown in Figure 2.4(c), the embedding we get is fairly good knowing that the data is not explainable by a shape basis. The periodicity of the motion is obvious as several tracks overlap.

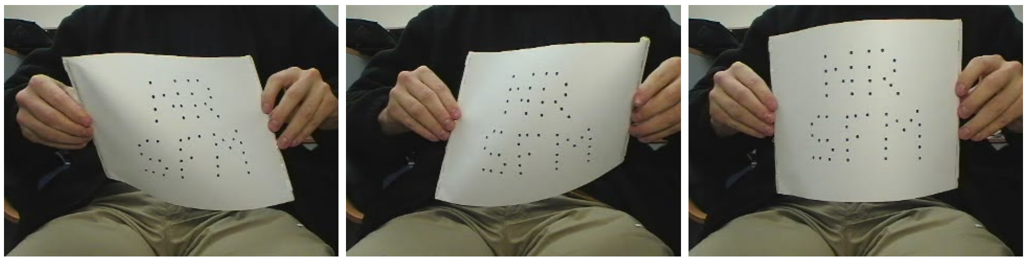


(a) Sampling

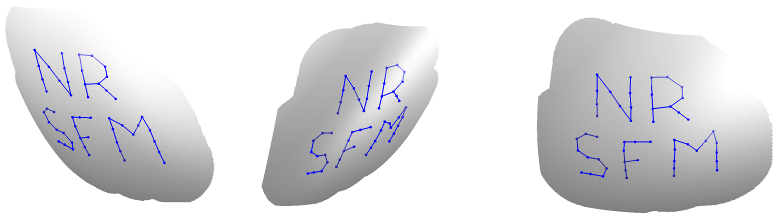


(b) Recovered Embedding

Figure 2.5: **Bending Sheet Dataset** Figure (a) illustrates the sampling as described in Figure 2.2(a). We can notice the periodicity of the motion with the different green descending diagonals. On the other hand, the ascending green diagonals illustrates the fact that when reaching an end, it looks equivalent to having the sequence going forward or backward in time. This is even more obvious when looking at the recovered manifold in Figure (b). The sheet is twisted between two extremes. When it goes from state yellow to state light blue (frames 18 and 44), it goes from one extreme position to another: hence their extreme position on the manifold. On the other hand, similar positions (like light blue, blue and darkest gray) according to the sampling matrix, will reflect close-by points on the manifold. Finally, the black point and its neighbor do not belong to an end of the manifold as the sheet was not twisted fully (leading to some horizontal lines with few green/similarities around rows 70-80).



(a) Sample Frames from the Sheet video sequence



(b) Reconstructions corresponding to the above frames, rendered from novel view points

Figure 2.6: Figure (a) shows 3 frames from the original sequence where the sheet is twisted. In Figure (b), three reconstructed views are presented under a novel view point: a quadratic surface was fit through the points and illuminated so as to highlight the bending.

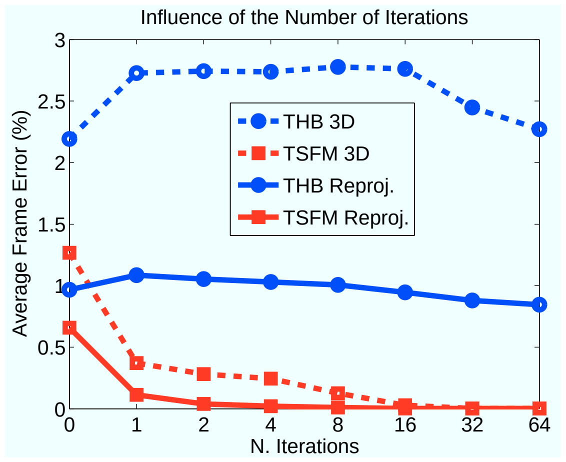


Figure 2.7: **Importance of the Number of Iterations.** The reprojection error (full line) and the 3D reconstruction error (dashed) are reprensented for the THB algorithm and our approach. The iterations for THB correspond to the EM iterations while the ones for TSFM are the ones for the gradient descent.

2.6 Pushing the Limits

2.6.1 Gradient Descent Iterations

Figure 2.7 demonstrates the (un-)importance of iterations in TSFM. While the original estimate is worse than THB (both in reprojection and 3D error), it only requires one iteration of gradient descent to be better, and a few more to reach a very small reconstruction error. After 50 iterations, the 3D reconstruction error is also virtually null (we consider 10^{-27} as null).

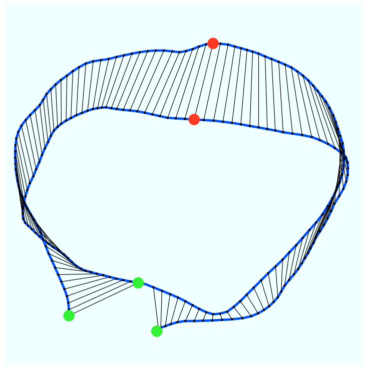
This can be explained by the fact that the original estimate is more a loose estimate than a close estimate: the obtained embedding and variables are not perfect but are similar enough to the true one that gradient descent seems to converge to the global optimum. The EM optimization of THB seems slower and also only seems to affect the reprojection error, hence leading to a wrong reconstruction.

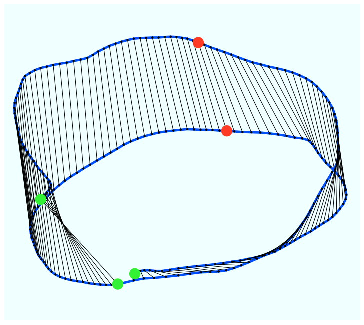
2.6.2 Triplet/Pair Sampling

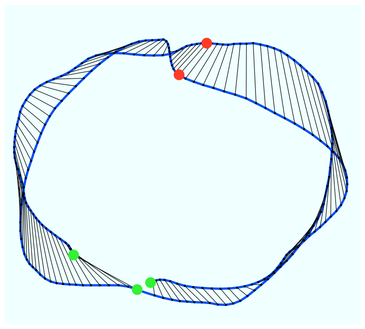
Figure 2.8 demonstrates the importance of sampling when computing the initial shape embedding of the shark data. As expected, the more samples, the better the embedding. Unfortunately, the computation increases in a non-linear way, and the usual compromise between accuracy and speed arises.

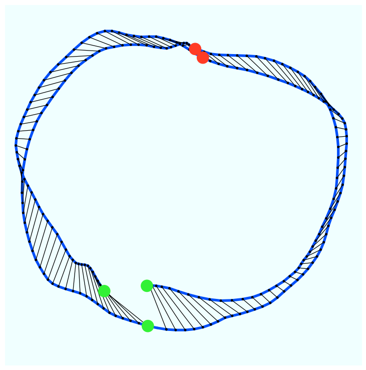
2.6.3 Noise Robustness

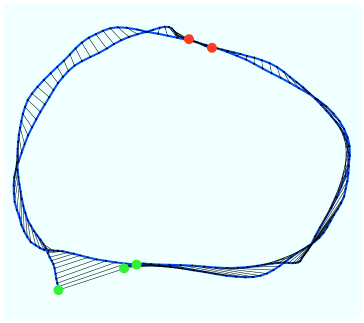
Figure 2.9 illustrates the robustness of out method to noise. For low levels of noise, our method performs better than state of the art: this is due to the fact that recovering the embedding is not really sensitive to the noise, as all that is needed is a loose approximation. It is worth noting that TSFM seems more usntable than THB as the results it give (though better in average) have a larger standard deviation. This is probably due to the low sampling as explained in Section 2.6.2


(a) 1×240


(b) 2×240


(c) 4×240


(d) 6×240


(e) 8×240

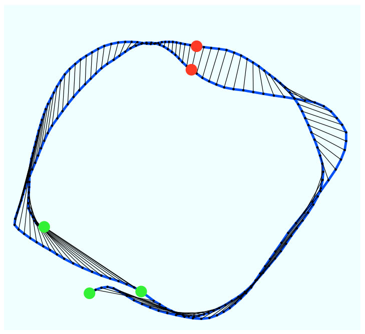
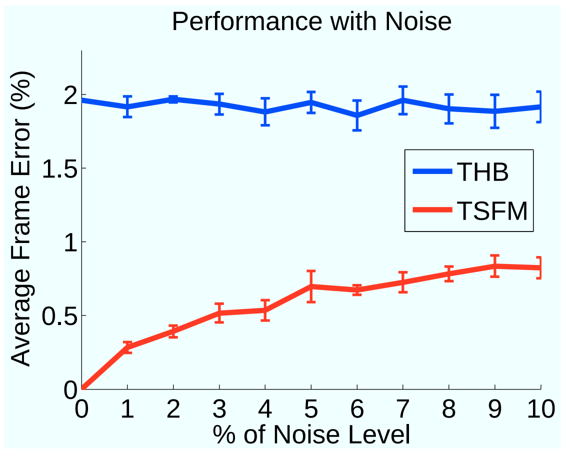
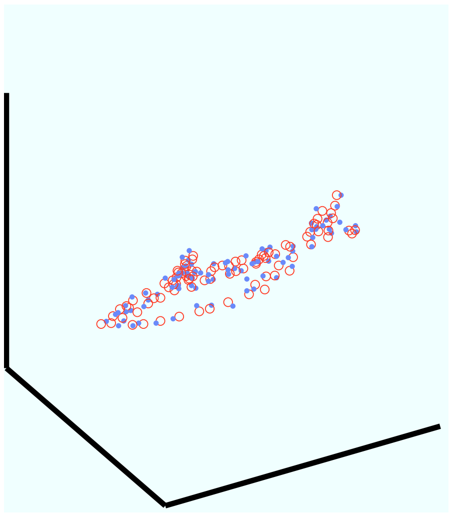

(f) 14×240

Figure 2.8: **Importance of the Triplet/Pair Sampling.** These different figures show the embeddings constructed by the first part of our algorithm. The video sequence considered is the shark one and different amount of samples are chosen (They are all multiple of the length of the sequence: 240 frames). Below each figure is indicated the number of triplet sample chosen to compute the embedding (the number of pairs is chosen to be the same).



(a) Performance with Noise



(b) Frame with 10% of noise

Figure 2.9: **Robustness to Noise** Figure (a) displays the performance of THB and TSFM with respect to noise (XCK was excluded because of its poor performance). The amount of noise (in percentage) is $\|err\| \,/\, \|W\|$. As an example of how much noise is 10%, we display a frame in Figure (b).

2.7 Acknowledgements

Chapter 2, is a reprint of the material as it appears in Linear Embeddings in Non-Rigid Structure from Motion, V. Rabaud and S. Belongie, IEEE Conference on Computer Vision and Pattern Recognition, 2009.

Chapter 3

LSML

In the previous section, we showed a new approach to NRSFM that competes with state of the art techniques but that still uses a linear model for the shape manifold.

While this method can be computationally efficient and well-suited to common objects of study (*e.g.* faces), there is no reason to believe that the possible 3D shapes of an object lie on a linear low-dimensional manifold (*cf*.Figure 3.1).

If we relax this assumption by assuming that only small neighborhoods of shapes are well-represented by a linear subspace, the set of possible 3D shapes can now be described as a smooth and low-dimensional manifold. Also, as a local neighborhood contains the different instances of how a 3D shape can deform, its dimensionality (and therefore, that of the manifold) is the number of degrees of freedom of the object.

With such an interpretation, a new manifold technique needs to be developed in order to focus on the tangents of the manifold, and not the points themselves. This chapter is dedicated to presenting LSML (*Locally Smooth Manifold Learning*), a method I developed with Piotr Dollár to learn the tangent field of a manifold

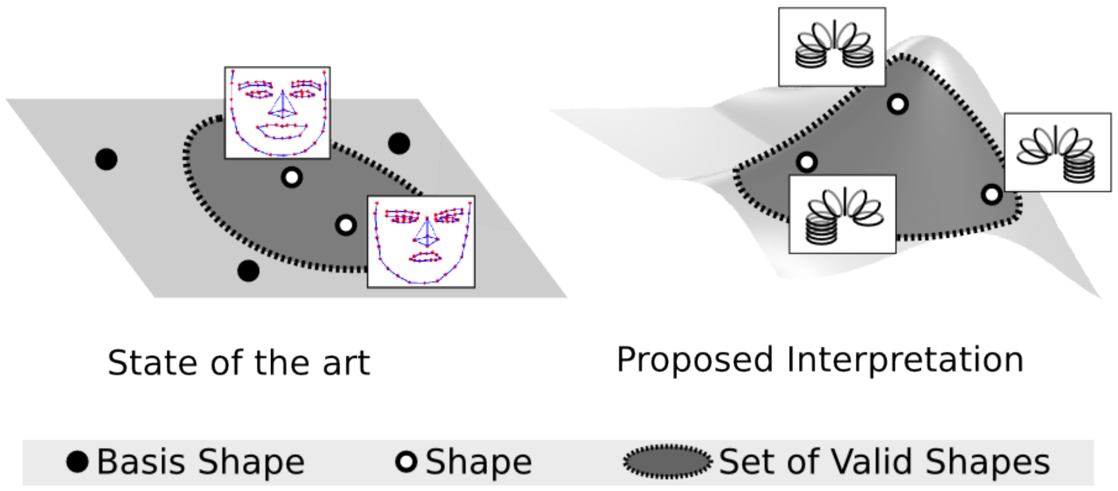


Figure 3.1: State of the art in non-rigid structure from motion assumes that a deformable 3D shape can be expressed as a linear combination of basis shapes. While this is a well-studied Cootes et al. (2001) and convenient assumption (*e.g*. for faces), there is no reason to believe that the manifold of the possible shapes of an object is linear (*e.g*., as we will discover further, it is highly non-linear for a Slinky®/spring toy). In this work, the only constraint imposed on the 3D shape manifold is its dimensionality.

3.1 Related Work

3.1.1 Common Manifold Learning Techniques

A number of methods have been developed for dealing with high dimensional data sets that fall on or near a smooth low dimensional nonlinear manifold. Such data sets arise whenever the number of modes of variability of the data is much smaller than the dimension of the input space, as is the case for image sequences and objects deforming. Unsupervised manifold learning refers to the problem of recovering the structure of a manifold from a set of unordered sample points. Usually, the problem is formulated in terms of finding an embedding or "unrolling" of a manifold into a lower

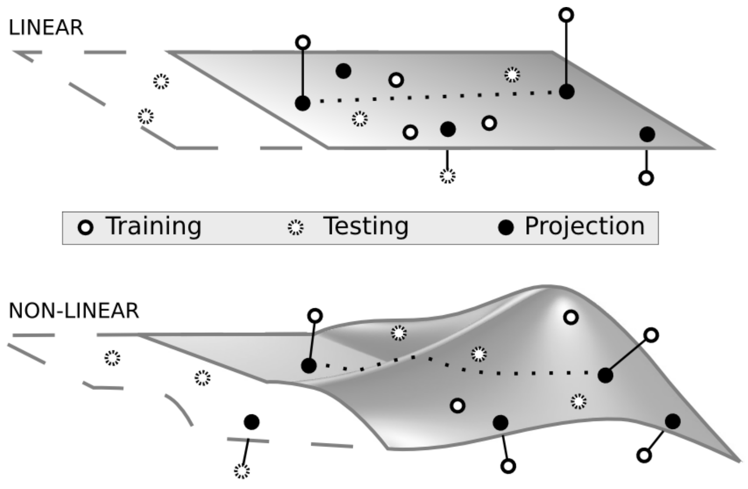


Figure 3.2: Given the linear subspace (manifold) found by PCA, it is straightforward to project points onto the manifold, measure the distance between a point and the manifold, measure distance between points after projection, and to predict the structure of the manifold in regions with little or no data. In this work we present an algorithm that treats manifold learning as a problem of generalizing to unseen portions of a manifold and can be used much like PCA but for nonlinear manifolds. Also, unlike nonlinear embedding methods, LSML can be used with non-isometric manifolds like the one above.

dimensional space such that certain geometric relationships between the original points are preserved, as in the seminal works of ISOMAP (Tenenbaum et al. (2000)) and (LLE Roweis and Saul (2000)).

Although good embedding results have been obtained for a number of manifolds, embedding methods are limited in two fundamental ways. First, they are by definition well suited only for manifolds that are *isometric* (or conformal) to a subset of Euclidean space. There is little reason to believe that many manifolds of interest have this property, *e.g.* a sphere does not. Secondly, embedding methods are designed to describe a fixed set of data and not to *generalize* to novel data. Although out-of-sample extensions have been proposed (Bengio et al. (2004)), these are typically applicable only in regions of a manifold that have already been sampled densely. Such methods cannot be used to predict manifold structure where no samples are given, even if the manifold has a smooth, consistent form.

Consider classical MDS, used to find a distance preserving embedding from a matrix of Euclidean distances, and PCA, which finds a low dimensional subspace that explains observed data (Hastie et al. (2001)). Although both are designed to deal with linear subspaces, classical MDS is used primarily for data visualization and exploration while PCA is used for finding a low dimensional subspace that can be used to analyze novel data. PCA tends to be more versatile than MDS: given the linear subspace (manifold) found by PCA, it is straightforward to project points onto the manifold, measure the distance between a point and the manifold, measure distance between points after projection, and to predict the structure of the manifold in regions with little or no data.

Nonlinear embedding methods are used much like MDS but for nonlinear manifolds; however, they cannot be used for manipulating novel data. In this work we present an extension of LSML Dollár et al. (2006), and show that it can be used much like PCA but for nonlinear manifolds (see Figure 3.2). LSML treats manifold learning as a problem of generalizing to unseen portions of a manifold, and is applicable to non-isometric cases. Instead of only finding an embedding for visualization, LSML learns a representation of a manifold or family of related manifolds that can be used for computing geodesic distances, finding the projection of a point onto a manifold, recovering a manifold from points corrupted by noise, generating novel points on a manifold, and more.

3.1.2 Previous Work Overview

Traditional methods of nonlinear manifold learning include self organizing maps, principal curves, and variants of multi-dimensional scaling (MDS) among others, see Hastie et al. (2001) for a brief introduction to these methods. In recent years, the field has seen a number of interesting developments. Schölkopf et al. (1998) introduced a kernelized version of PCA. A number of related embedding methods have also been introduced, representatives include LLE (Saul and Roweis, 2003), ISOMAP (Tenenbaum et al., 2000), h-LLE (Donoho and Grimes, 2003), and more recently MVU (Weinberger and Saul, 2006). Broadly, such methods can be classified as spectral embedding methods (Weinberger and Saul, 2006); the embeddings they compute are based on an eigenvector decomposition of an $n \times n$ matrix that represents geometrical relationships of some form between the original n points. Out-of-sample extensions have been proposed in Bengio et al. (2004).

Four methods that LSML shares inspiration with are (Brand, 2003; Keysers et al., 2001; Bengio and Monperrus, 2005; Rao and Ruderman, 1999). Brand (2003) employs a novel charting based method to achieve increased robustness to noise and decreased probability of pathological behavior *vs*. LLE and ISOMAP; we exploit similar ideas in the construction of LSML but differ in motivation and potential applicability. Bengio and Monperrus (2005) proposed a method to learn the tangent space of a manifold and demonstrated a preliminary illustration of rotating a small bitmap image by about 1°. Although our approach to the problem differs, the motivation for LSML is based on similar insights. Work by Keysers et al. (2001) is based on the notion of learning a model for class specific variation, the method reduces to computing a linear tangent subspace that models variability of each class. Rao and Ruderman (1999) shares one of our goals as it addresses the problem of learning Lie groups, the infinitesimal generators of certain geometric transformations.

Finally, there is a vast literature on computing distances between objects undergoing known transformations Miller and Younes (2001); Simard et al. (1998), which is essentially the problem of computing distances between manifolds with *known* structure.

3.2 LSML

Here we give details of LSML. For an introduction to LSML see Figure 3.3.

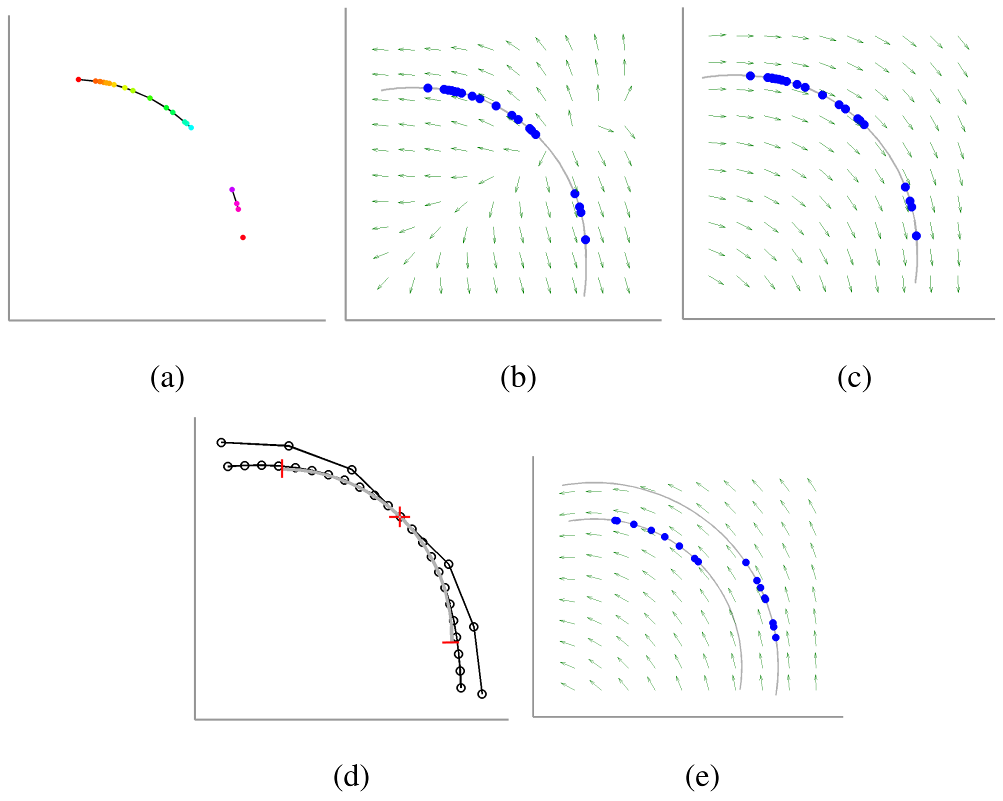


Figure 3.3: **Overview**. Twenty points (n=20) that lie on 1D curve (d=1) in a 2D space (D=2) are shown in (a). Black lines denote neighbors, in this case the neighborhood graph is not connected. We apply LSML to train $\mathcal{H}$ (with $f = 4$ RBFs). $\mathcal{H}$ maps points in $\mathbb{R}^2$ to tangent vectors; in (b) tangent vectors computed over a regularly spaced grid are displayed, with original points (blue) and curve (gray) overlayed. Tangent vectors near original points align with the curve, but note the seam through the middle. Regularization fixes this problem (c), the resulting tangents roughly align to the curve along its entirety. We can traverse the manifold by taking small steps in the direction of the tangent; (d) shows two such paths, generated starting at the red plus and traversing outward in large steps (outer curve) and finer steps (inner curve). In (e) two parallel curves are shown, with n=8 samples each. Training a common $\mathcal{H}$ results in a vector field that more accurately fits each curve than training a separate $\mathcal{H}$ for each (if their structure was very different this need not be the case).

3.2.1 Motivation and Error Function

Let D be the dimension of the input space, and assume the data lies on a smooth d-dimensional manifold ($d \ll D$). For simplicity assume that the manifold is diffeomorphic to a subset of $\mathbb{R}^d$, meaning that it can be endowed with a global coordinate system (this requirement can easily be relaxed) and that there exists a continuous bijective mapping $\mathcal{M}$ that converts coordinates $\mathbf{y} \in \mathbb{R}^d$ to points $\mathbf{x} \in \mathbb{R}^D$ on the manifold. The goal is to learn a warping function $\mathcal{W}$ that can take a point on the manifold and return any neighboring point on the manifold, capturing all the modes of variation of the data.

Define $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon}) = \mathcal{M}(\mathbf{y} + \boldsymbol{\epsilon})$, where $\mathbf{y} = \mathcal{M}^{-1}(\mathbf{x})$ and $\boldsymbol{\epsilon} \in \mathbb{R}^d$. Taking the first order approximation of the above gives: $\mathcal{W}(\mathbf{x}, \boldsymbol{\epsilon}) \approx \mathbf{x} + \mathcal{H}(\mathbf{x})\boldsymbol{\epsilon}$, where each column $\mathcal{H}_{\cdot k}(\mathbf{x})$ of the matrix $\mathcal{H}(\mathbf{x})$ is the partial derivative of $\mathcal{M}$ w.r.t. $\mathbf{y}_k$: $\mathcal{H}_{\cdot k}(\mathbf{x}) = \partial/\partial \mathbf{y}_k \mathcal{M}(\mathbf{y})$. This approximation is valid given $\boldsymbol{\epsilon}$ small enough.

The goal of LSML is to learn the function $\mathcal{H}_\theta : \mathbb{R}^D \rightarrow \mathbb{R}^{D \times d}$ parameterized by a variable θ. Only data points $\mathbf{x}^i$ sampled from one or several manifolds are given. For each $\mathbf{x}^i$, the set $\mathcal{N}^i$ of neighbors is then computed (*e.g.* using variants of nearest neighbor such as kNN or ϵNN), with the constraint that two points can be neighbors only if they come from the same manifold. The original formulation of LSML was based on the observation that if $\mathbf{x}^j$ is a neighbor of $\mathbf{x}^i$, there then exists an *unknown* $\boldsymbol{\epsilon}^{ij}$ such that $\mathcal{W}(\mathbf{x}^i, \boldsymbol{\epsilon}^{ij}) = \mathbf{x}^j$, or to a good approximation:

$$\mathcal{H}_\theta(\mathbf{x}^i)\boldsymbol{\epsilon}^{ij} \approx \Delta^i_{\cdot j}, \tag{3.1}$$

where $\Delta^i_{\cdot j} \equiv \mathbf{x}^j - \mathbf{x}^i$. An interpretation of the above is that $\Delta^i_{\cdot j}$ is the *un-centered* estimate of a directional derivative at $\mathbf{x}^i$. However, $\Delta^i_{\cdot j}$ could also serve as the *centered* estimate of the directional derivative at $\overline{\mathbf{x}}^{ij} \equiv \frac{\mathbf{x}^i + \mathbf{x}^j}{2}$:

$$\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} \approx \Delta^i_{\cdot j}. \tag{3.2}$$

$\overline{\mathbf{x}}^{ij}$ may lie slightly off the manifold, however, as $\mathcal{H}_\theta$ is a smooth mapping over all of $\mathbb{R}^D$, this does not pose a problem. Although the change is subtle, in practice use of (3.2) provides significant benefit, as the centered approximation of the derivative has no second order error. So, roughly speaking (3.1) is valid if locally the manifold has a good linear approximation while (3.2) is valid where a quadratic approximation holds.

To solve for $\mathcal{H}_\theta$, we define the following error:

$$\text{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j\in\mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j}\right\|_2^2. \tag{3.3}$$

We want to find the θ that minimizes this error. The $\boldsymbol{\epsilon}^{ij}$ are additional free parameters that must be optimized over; they do not affect model complexity.

3.2.2 Regularization

We can explicitly enforce the mapping $\mathcal{H}_\theta$ to be smooth by adding a regularization term (in addition to implicit smoothness that may come from the form of $\mathcal{H}_\theta$ itself). For each i, the learned tangents at two neighboring locations $\overline{\mathbf{x}}^{ij}$ and $\overline{\mathbf{x}}^{ij'}$ should be similar, *i.e.* $\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'})\|_F^2$ should be small. Note that this may not always be possible, *e.g.* the Hairy Ball Theorem states there is no non-trivial smooth vector field on a sphere. To ensure that the $\mathcal{H}_\theta$'s do not get very small and the $\boldsymbol{\epsilon}$'s very large, $\|\boldsymbol{\epsilon}^{ij}\|_2^2$ must also be constrained. We add the following term to (3.3):

$$\lambda_E \sum \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 + \lambda_\theta \sum \left\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'})\right\|_F^2. \tag{3.4}$$

The overall error can be rewritten using a single λ if for any scalar $a > 0$ we treat $\mathcal{H}_\theta$ and $a\mathcal{H}_\theta$ as essentially the same. The error of $\mathcal{H}_\theta$ with regularization parameters $(\lambda_E, \lambda_\theta)$ is the same as the error of $a\mathcal{H}_\theta$ with regularization parameters $(a^2\lambda_E, \frac{1}{a^2}\lambda_\theta)$. Thus there is a single degree of freedom, and we always set $\lambda_E = \lambda_\theta = \lambda$.

3.2.3 Linear Parametrization

Although potentially any regression technique is applicable, a linear model is particularly easy to work with. We use radial basis functions (RBFs) to define additional features for the data points Hastie et al. (2001). The number of basis functions, f, is an additional parameter that controls the smoothness of the final mapping $\mathcal{H}_\theta$. Let $\mathbf{f}^{ij}$ be the f features computed from $\overline{\mathbf{x}}^{ij}$. We can then define $\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) = \left[\Theta^1\mathbf{f}^{ij} \cdots \Theta^D\mathbf{f}^{ij}\right]^\top$, where each Θ^k is a $d \times f$ matrix. Re-arranging (3.3) gives:

$$\text{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j\in\mathcal{N}^i} \sum_{k=1}^{D} \left(\mathbf{f}^{ij\top}\Theta^{k\top}\boldsymbol{\epsilon}^{ij} - \Delta^i_{kj}\right)^2. \tag{3.5}$$

Solving simultaneously for the $\boldsymbol{\epsilon}$'s and Θ's is complex, but if either the $\boldsymbol{\epsilon}$'s or Θ's are fixed, solving for the remaining variables becomes a least squares problem (details omitted for space). We use an alternating minimization procedure, with random restarts to avoid local minima.

3.2.4 Radial Basis Functions

For the features we use radial basis functions (RBFs) Hastie et al. (2001), the number of basis functions, f, being an additional parameter. Each basis function is of the form $f^j(\mathbf{x}) = \exp(-\|\mathbf{x} - \boldsymbol{\mu}^j\|_2^2/2\sigma^2)$ where the centers $\boldsymbol{\mu}^j$ are obtained using K-means clustering on the original data with f clusters and the width parameter σ is set to be twice the average of the minimum distance between each cluster and its nearest neighbor center. The feature vectors are then simply $\mathbf{f}^i = [f^1(\mathbf{x}^i) \cdots f^p(\mathbf{x}^i)]^\top$. The parameter f controls the smoothness of the final mapping $\mathcal{H}_\theta$; larger values result in mappings that better fit local variations of the data, but whose generalization abilities to other points on the manifold may be weaker (see Section 3.3.2).

3.3 Analyzing Manifold Learning Methods

In this section we seek to develop a methodology for analyzing nonlinear manifold learning methods without resorting to subjective visual assessment of the quality of an embedding. The motivation is twofold. First, we wish to have an objective criterion by which to compare LSML to existing manifold learning methods. Second, and more importantly, in order to work with non-isometric manifolds (which may not have meaningful embeddings), we need to establish some guidelines for how to properly control the complexity and evaluate the performance of a manifold learning method.

A key issue in supervised learning is the generalization performance of a learning method, defined by its prediction error computed on independent test data (Hastie et al. (2001)). Such a notion is of central importance for supervised learning because it provides a principled approach for choosing among learning methods, controlling their complexity, and evaluating their performance. We will show how some of these ideas can be applied in the context of nonlinear manifold learning.

3.3.1 Model Evaluation

A number of works have established the theoretical relationships between various manifold embedding methods Bengio et al. (2004); Xiao et al. (2006b). Also, asymptotic guarantees exist for ISOMAP, h-LLE and MVU, and conceivably similar bounds could be shown for other methods. Here, however, we are more interested in how these methods perform with a finite sample. We begin by introducing a simple yet intuitive evaluation methodology.

By definition, if a manifold is isometric to a convex subset of a low dimensional Euclidean space, there exists a diffeomorphism that maps the metric of such a manifold, defined by geodesic distance, to the Euclidean metric. A natural measure of the quality of an embedding for this class of manifolds is how closely distance is preserved[1]. Given a sufficient number of samples from an isometric manifold, geodesic distance can be estimated accurately, and this estimate is guaranteed to converge as the number of samples grows (this forms the basis for ISOMAP, we refer the reader to Tenenbaum et al. (2000) for details). We evaluate finite sample performance, using a much larger sample to obtain ground truth distances. This methodology is applicable for manifolds that can be sampled densely, for example for toy data or for image manifolds where we know the underlying transformation or generative model for the manifold.

[1]LLE, MVU and LSML are applicable to larger classes of manifolds. However, any method should work with isometric manifolds. There have also been some results on conformal embeddings (which preserve angles but not distances); a similar methodology could be developed for this case. Finally, note that LLE finds an embedding that can differ by an affine transformation from a distance preserving embedding, and so must be adjusted accordingly.

We assume we are given two sets of samples from the manifold, S_n containing n points, which serves as the training data, and a very large set S_∞, used only during the evaluations stage. The ground truth geodesic distances d_{ij} for each pair of points i, j in S_n are computed using S_∞. Let d'_{ij} denote the Euclidean distance between points i, j in a given embedding. We define the error of an embedding as:

$$\text{err}_{\text{GD}} \equiv \frac{1}{n^2} \sum_{ij} \frac{|d_{ij} - d'_{ij}|}{d_{ij}}. \tag{3.6}$$

All experiments presented here were averaged over 10 trials. Reasonable effort was made to maximize the performance of each method tested. We compared LSML to three embedding methods: ISOMAP Tenenbaum et al. (2000), LLE Roweis and Saul (2000) and (fast) MVU Weinberger and Saul (2006), each of which has code available online. The embedding methods require a fully connected neighborhood graph; we simply discarded data that resulted in disconnected graphs. We sampled at most 1000 pairs of neighboring points to train LSML, under these conditions training takes around two minutes regardless of n or the number of neighbors k. Details of how to use the learned model for geodesic distance computation are given in Section 3.4.3. Since computing pairwise distances is time consuming for large n, we sample 100 landmark points.

The first experiments examine finite sample performance (see Figure 3.4). The performance of the embedding methods ranked as ISOMAP $\succ$ MVU $\succ$ LLE, with the performance of LLE being quite bad even for large n. To be fair however, LLE can recover embeddings for classes of manifolds where MVU and ISOMAP cannot (and likewise MVU is more general than ISOMAP). LSML performed better than these embedding methods, especially for small n. LSML-U refers to the version of LSML from Dollár et al. (2006), it generally performs similarly to LSML, except for small n or large k (when curvature is non-negligible).

3.3.2 Model Complexity

The first step of many embedding methods is to construct a neighborhood graph of the data, using for example the k-nearest neighbors of each point. Thus all these

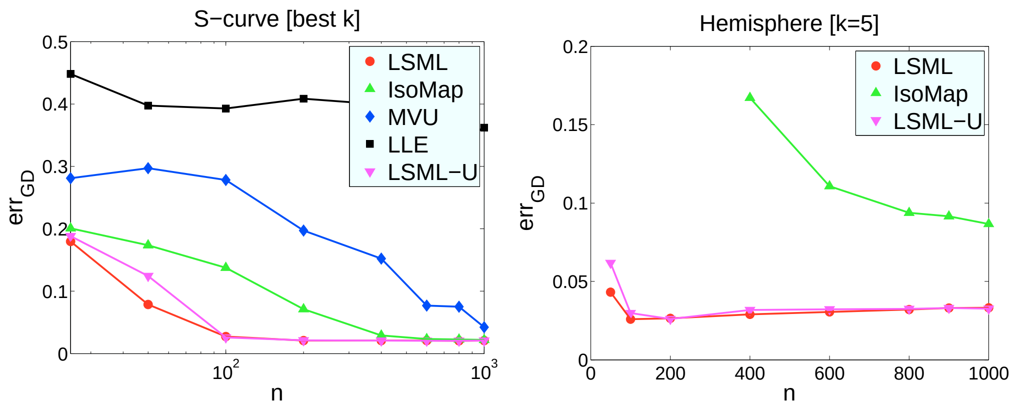


Figure 3.4: **Finite sample performance**. Performance of various algorithms, measured by $\mathrm{err}_{\mathrm{GD}}$ as n is varied. All results shown are averaged over 10 repetitions. *Left*: S-curve error for various methods (k chosen optimally for each data point). ISOMAP, MVU and LSML all performed well for large n. LSML performed quite well even with n=25. Qualitatively similar results were obtained for other manifolds. *Right*: ISOMAP (geodesic distance computation only) and LSML results on a hemisphere. Here $k = 5$ was fixed, preventing ISOMAP from converging to a lower error solution. LSML's performance did not improve after n=100 because of sampling (see text).

methods have at least one parameter, k, usually set by hand. If k is too small estimates of local geometry may have high error or *variance*, either due to noise or lack of information. If k is too large estimates of local geometry may be *biased* by manifold curvature. For the trivial case of a linear subspace, which has no curvature, increasing k should continuously decrease error since there is no bias term.

In these terms the choice of k is reminiscent of the classic bias-variance tradeoff in supervised learning. Another view of the tradeoff is that enlarging k increases the number of constraints on the embedding, except that additional constraints become increasingly inaccurate. See Figure 3.5 for the effects of neighborhood size and noise on performance for each method.

In addition to k, LSML has two more smoothing parameters: the number of RBF's and the regularization parameter λ. LLE also has a regularization term whose

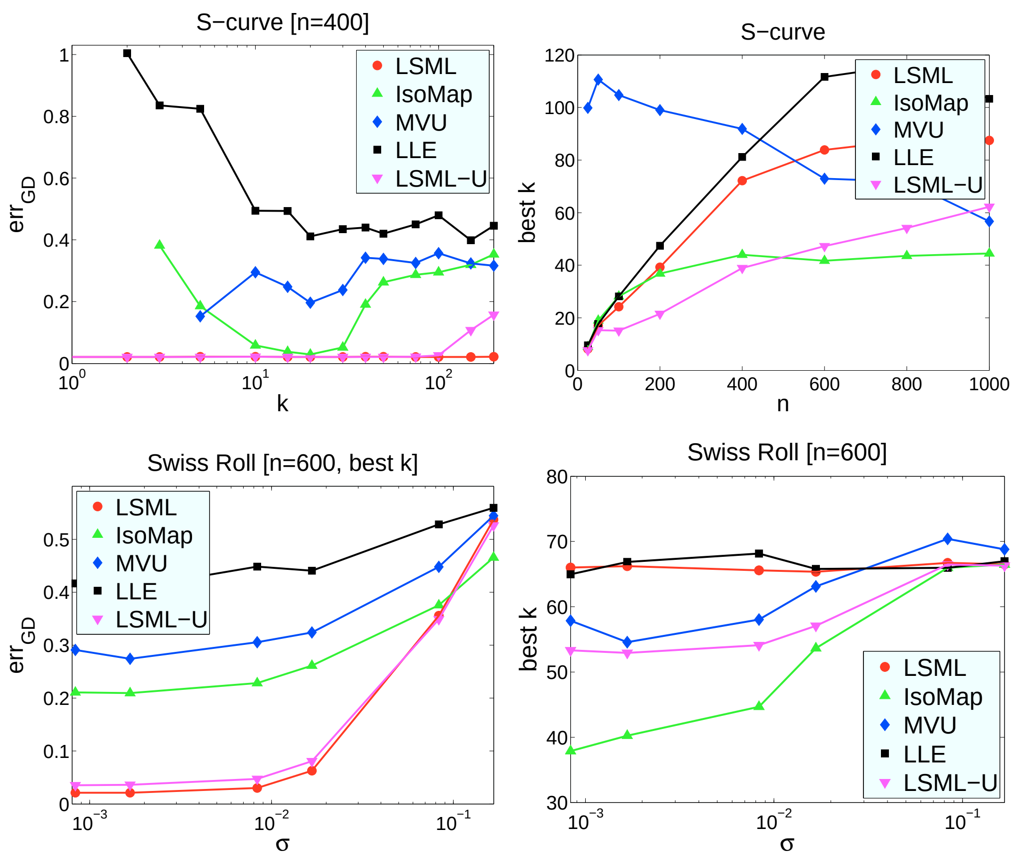


Figure 3.5: **Bias-Variance Tradeoff**. *Top*: Effects of neighborhood size. *Left*: Error as a function of k for n=400 points from S-curve; note 'U'-shaped curves. Note also the robustness of LSML to very small (k=1) and very large (k=100) neighborhood sizes (eventually LSML does break down, not shown). The reason for this robustness is due to use of the centered error (LSML-U is not quite as robust for large k), see Section 3.2.1. *Right*: Best value of k for each method as a function of n. For all but MVU k increases sub-linearly with n. *Bottom*: Effects of noise. *Left*: As the amount of noise σ increases, the error of each method increases. LSML performs well, even without the de-noising introduced in Section 3.4.2. *Right*: Best value of k for each method as a function of σ. For MVU and ISOMAP k increases to compensate for additional noise, LSML always prefers fairly large k given noise.

effects can be significant and MVU has a parameter ω that balances between maximizing variance and penalizing slack. Ideally, we would like some way of automatically performing model selection; *i.e.* automatically selecting the value for each of these parameters for a given problem instead of relying on a visual assessment of the final embedding.

3.3.3 Model Selection

err_{GD} can be used to analyze model complexity in certain settings, as above. However, in general there is no large sample set available from which to compute ground truth distances, in which case err_{GD} cannot be used to perform model selection. Also, for the same reasons that in general the quality of an embedding cannot be evaluated, neither can the quality of out-of-sample extension, so this does not provide an answer[2]. The key difficulty stems from the fact that most manifold embedding methods make no *testable* predictions.

If manifold learning can be posed in such a way that it makes testable predictions, the quality of the predictions can potentially be used to perform model selection. For example, if in addition to an embedding the reverse mapping back to $\mathbb{R}^D$ is also learned, error can be measured by how well a test point is encoded. This principle formed the basis for auto-encoding neural nets DeMers and Cottrell (1993).

Given a notion of test error, standard techniques from supervised learning, such as cross-validation, can be used for performing model selection and evaluation for manifold learning. However, although any testable prediction gives rise to an error measure, not all error measures are necessarily useful. err_{GD} gives a way to evaluate performance in certain scenarios; at minimum prediction error should roughly correlate with err_{GD} when it is available.

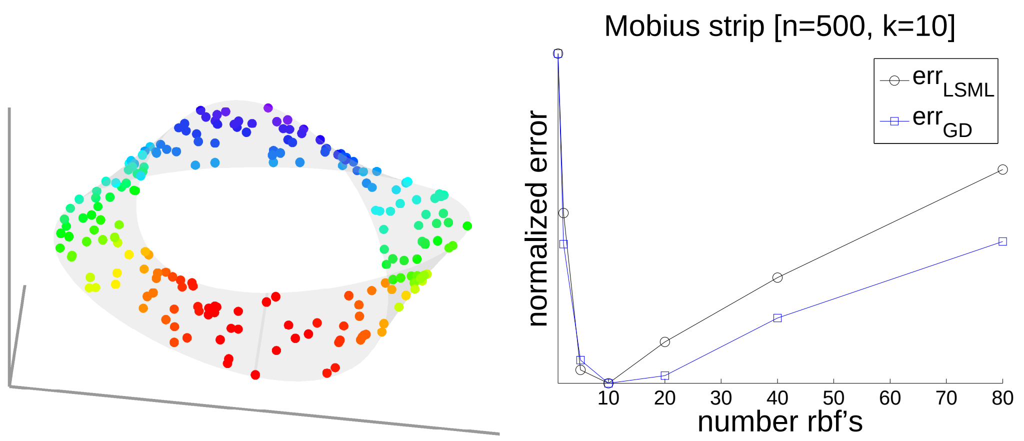


Figure 3.6: **LSML Test Error**. When err_{GD} can be computed it is strongly correlated with err_{LSML}, supporting our use of err_{LSML} to perform model evaluation. *Left*: Points drawn from a Möbius strip, a manifold with $d = 2$, with a small amount of noise added. *Right*: err_{LSML} and err_{GD}, each normalized to lie between 0 and 1, as the number of RBFs was varied. Of 500 points, $\frac{2}{3}$ were used for training, $\frac{1}{3}$ for testing. According to both error measures approximately 10 to 20 RBFs was best.

3.3.4 LSML Test Error

LSML predicts the tangent planes $\mathcal{H}$ at all points on a manifold. We can define the test error of LSML according to how well the predicted tangent planes fit unseen data (see Eq. (3.2)). Given unseen test points $\mathbf{x}^i$, the error is:

$$\text{err}_{\text{LSML}} \equiv \sum_i \min_{\boldsymbol{\epsilon}^{ii'}} \left\| \mathcal{H}_\theta(\overline{\mathbf{x}}^{ii'})\boldsymbol{\epsilon}^{ii'} - (\mathbf{x}^i - \mathbf{x}^{i'}) \right\|_2^2, \tag{3.7}$$

where $\mathbf{x}^{i'}$ is the closest neighbor of $\mathbf{x}^i$, either another test point or a training point.

err_{LSML} is strongly correlated with err_{GD} both for isometric and non-isometric manifolds. For example, in Figure 3.6 we show the effect of changing the number of RBFs on both errors for data that lies on a Möbius strip; note that the minima of the errors occur at roughly the same spot. This correlation is important, because it allows us to use err_{LSML} in place of err_{GD} to perform model evaluation, select model parameters,

[2] Bengio et al. (2004) evaluated the out-of-sample extension of various methods by seeing how consistent the out-of-sample prediction was compared to simply including the point in the original embedding. However this is only a partial solution since consistency does not imply accuracy (consider a trivial embedding which maps all points to the origin).

etc. Although we can only prove the utility of err_{LSML} for manifolds for which err_{GD} is defined, err_{LSML} is most useful when err_{GD} cannot be computed.

Finally, note that err_{LSML} cannot be used to choose d, since as d increases err_{LSML} will decrease. This is the same challenge as choosing k in K-means clustering. A simple rule of thumb is to choose the smallest value of d such that further increasing d results in only a small decrease in error.

3.4 Working with $\mathcal{H}_\theta$

We now develop a number of uses for the representation $\mathcal{H}_\theta$ learned by LSML, including projection, manifold de-noising, geodesic distance computation, and manifold transfer. For all experiments we use err_{LSML} to select model parameters, including the number of RBF's, k, and the regularization parameter λ. Given low test error, we expect geodesic distance computation, de-noising, *etc*. to be accurate (see Section 3.3.4).

$\mathcal{H}_\theta$ is a function defined everywhere on $\mathbb{R}^D$, not just for points from the original manifold. This allows us to work with points that don't lie precisely on the manifold. Finally, note that in addition to $\mathcal{H}_\theta$ at least one training point is necessary to identify a manifold, and in practice keeping more training points is useful.

3.4.1 Projection

The projection of a point onto a linear subspace is unique and can be computed in closed form. For nonlinear manifolds this is in general not the case. $\mathbf{x}'$ is an orthogonal projection of a point $\mathbf{x}$ onto a manifold if it minimizes $||\mathbf{x} - \mathbf{x}'||_2^2$ in an open neighborhood. We can find such a point by initially setting $\mathbf{x}'$ to be some point from the manifold and then performing gradient descent, being careful not to leave the support of the manifold. $\mathbf{x}'$ is therefore bound to only follow the projection of the gradient on the local tangent space defined by $\mathcal{H}_\theta(\mathbf{x}')$. Let $H' \equiv \text{orth}(\mathcal{H}_\theta(\mathbf{x}'))$ be the $d \times D$ matrix that denotes the orthonormalized tangent space at $\mathbf{x}'$, and $H'H'^\top$ the corresponding

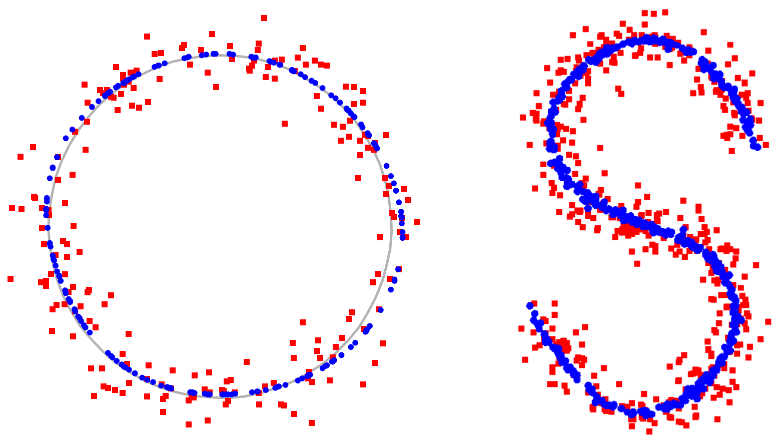


Figure 3.7: **Manifold De-noising**. Red (square) points are the noisy training data used to learn $\mathcal{H}_\theta$. After, the de-noising procedure from Section 3.4.2 was applied to recover the blue (circular) points. Original manifolds shown in gray. *Left*: Circle (D=2, d=1, n=200). *Right*: S-curve, side view (D=3, d=2, n=500).

projection matrix. The update rule for $\mathbf{x}'$ (with step size α) is:

$$\mathbf{x}' \leftarrow \mathbf{x}' + \alpha H' H'^\top (\mathbf{x} - \mathbf{x}'). \tag{3.8}$$

There may be multiple orthogonal projections but we generally are interested in the closest one. A simple heuristic is to initially set $\mathbf{x}'$ to the nearest point in the training data.

3.4.2 Manifold De-noising

Assume we have learned or are given the mapping $\mathcal{H}_\theta$ that describes the tangent space of some manifold or family of manifolds. Suppose we are also given samples $\mathbf{x}^i$ from a manifold that have been corrupted by noise so they no longer lie exactly on the manifold (these can be the same samples from which $\mathcal{H}_\theta$ was learned). Here we show how to recover a set of points χ^i that are close to the noisy samples $\mathbf{x}^i$ but lie on a manifold consistent with $\mathcal{H}_\theta$.

The key to the approach is the observation that, as in Eq. (3.3), if χ^j is a neighbor of χ^i then there exists an unknown $\boldsymbol{\epsilon}^{ij}$ such that $\|\mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j)\|_2^2$ is small. We

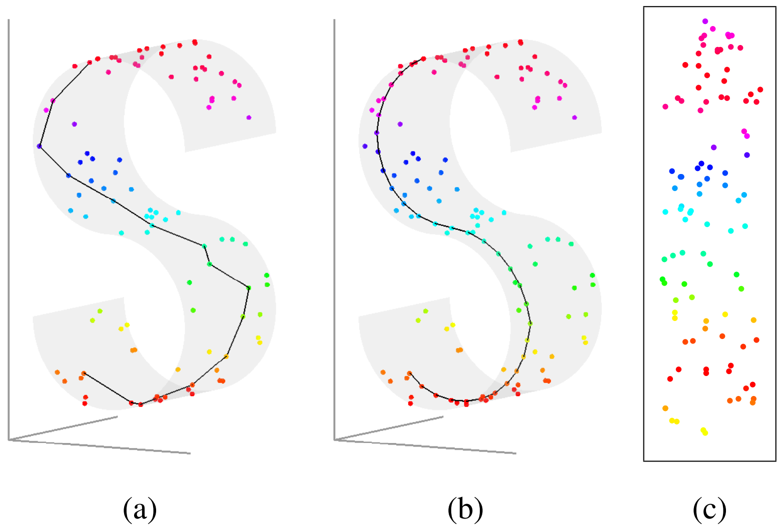

(a) (b) (c)

Figure 3.8: **Geodesic Distance**. We can find a geodesic path between two points by using a snake. The key idea is to start with a discretized path between $\mathbf{x}$ and $\mathbf{x}'$ and perform gradient descent until the length of the path cannot decrease. (a) Initial path between two points on S-curve. (b) Locally shortest path (geodesic) obtained after gradient descent. (c) Embedding computed using classical MDS on geodesic distances (only applicable for isometric manifolds). $\mathcal{H}_\theta$ was trained on the n=100 points shown.

can thus find a series of points χ^i that are close to the original points $\mathbf{x}^i$ and satisfy the above. The error to minimize is the sum of two terms:

$$\text{err}_{\mathcal{M}}(\chi) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j\in\mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j)\right\|_2^2 \tag{3.9}$$

$$\text{err}_{\text{orig}}(\chi) = \sum_{i=1}^{n} \left\|\chi^i - \mathbf{x}^i\right\|_2^2 \tag{3.10}$$

The second term is multiplied by a constant λ_{noise} to weigh the importance of sticking to the original points. We assume that for a small change in χ^i, $\frac{\partial\mathcal{H}_\theta}{\partial\chi^i}$ is negligible with respect to the other quantities. Under such an assumption, the gradient of $\text{err}_{\mathcal{M}}(\chi)$ can easily be rewritten as a product of matrices. We find a minimum energy solution by initially setting $\chi^i = \mathbf{x}^i$ for each i and then performing gradient descent on the χ^i's, this time using only the component of the gradient that is orthonormal to the manifold (we cannot correct the noise in the co-linear direction).

Manifold de-noising has received some attention in the literature Park et al. (2004); the approach presented here is simpler and does not suffer from the "trim the peak and fill the valley" phenomenon. We show results in Figure 3.7.

3.4.3 Geodesic Distance

Given two points on a manifold, $\mathbf{x}$ and $\mathbf{x}'$, we want to compute the geodesic (locally minimal) distance between them. To find such a minimal path we use an active contour model, also known as a 'snake' Kass et al. (1988); Blake and Isard (1998), where the length of a discretized path between $\mathbf{x}$ and $\mathbf{x}'$ is optimized by gradient descent.

Let $\chi^1 = \mathbf{x}$ and $\chi^m = \mathbf{x}'$, and $\chi^1, \ldots, \chi^m$ denote a sequence of points on the manifold forming a path between them. The length of the path is given by:

$$\sum_{i=2}^{m} \left\|\chi^i - \chi^{i-1}\right\|_2$$

For computational reasons, we use the following instead:

$$\text{err}_{\text{length}}(\chi) = \sum_{i=2}^{m} \left\|\chi^i - \chi^{i-1}\right\|_2^2 . \tag{3.11}$$

We minimize $\text{err}_{\text{length}}$ via gradient descent, keeping the ends of the snake fixed and again being careful not to leave the support of the manifold. The update rule for each χ^i is very similar to the update rule for projection given in Eq. (3.8).

To get an initial estimate of the snake, we apply Dijkstra's algorithm to the original points (also the first step of ISOMAP). To increase accuracy, additional points are linearly interpolated, and to ensure they lie on the manifold they are 'de-noised' by the procedure from Section (3.4.2) (with $\lambda_{\text{noise}} = 0$ and neighbors being adjacent points).

Figure 3.8 shows some example snakes computed over an S-curve. In Figure 3.9 we show an example snake that plausibly crosses a region where no points were sampled.

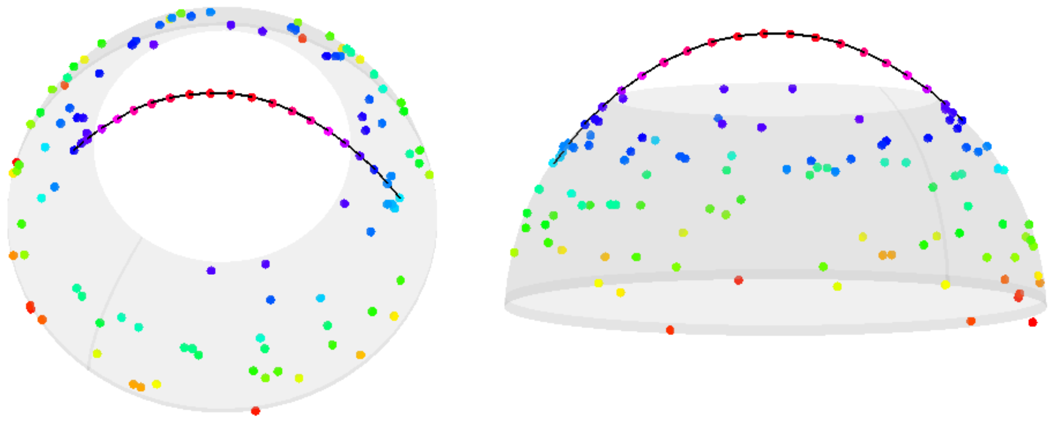


Figure 3.9: **Missing Data**. $\mathcal{H}_\theta$ was trained on n=100 points sampled from a hemisphere with the top portion removed. $\mathcal{H}_\theta$ plausibly predicted the manifold structure where no samples were given. Shown are two views of a geodesic computed between two points on opposite sides of the hole (see Section 3.4.3).

3.4.4 Manifold Transfer

A related problem to learning the structure of a single manifold is to simultaneously learn the representation of multiple manifolds that may share common structure (*cf*. Figure 3.3e). Such manifolds can arise whenever a common transformation acts on multiple objects, *e.g*. in the image domain this is quite common. One possibility is to relate representations learned separately for each manifold Elgammal and Lee (2004), however, learning simultaneously for all manifolds allows sharing of information.

An even more interesting problem is generalizing to *novel* manifolds. Given multiple sets of points, each sampled from a single manifold, we can formulate the problem as learning and generalizing to *unseen* manifolds. Once again err_{LSML} serves as the test error, except now it is computed from points on the unseen manifolds. LSML can already be trained on data of this form; two points being defined as neighbors if they are close in Euclidean space and come from the same manifold. Results of manifold transfer were shown in Figure 6 of Dollár et al. (2006).

3.5 Conclusion

In this chapter, we presented a novel examination of nonlinear manifold learning, posing it as the problem of learning manifold representations in a manner that allows for the manipulation of novel data. Drawing on concepts from supervised learning, we presented a theoretically sound methodology for quantifying the generalization performance of manifold learning approaches. With this formalism in hand, we presented LSML and showed how it can be applied to tasks including nonlinear projection, manifold de-noising, geodesic distance computation and manifold transfer.

The main feature of LSML we will now use for NRSFM is its manifold denoising property.

3.6 Acknowledgements

Chapter 3, is a reprint of the material as it appears in Non-Isometric Manifold Learning: Analysis and an Algorithm, P. Dollár, V. Rabaud and S. Belongie, International Conference on Machine Learning, 2007 and Learning to Traverse Image Manifolds, P. Dollár, V. Rabaud and S. Belongie, Neural Information Processing Systems, 2006.

Chapter 4

Non-Linear Embedding for NRSFM

In this chapter, we are moving away from the linear shape manifold assumption: just like in the previous chapter, the only constraint imposed on the shape manifold is its dimensionality. This evolution of the model is “natural” in the sense that it happened to other areas of computer vision. For example, shape models moved away from PCA/linear basis developed in Cootes et al. (2001) to non-linear models like in Tenenbaum and Freeman (2000) or Elgammal and Lee (2004).

By moving away from the linear basis interpretation and adopting a manifold-learning framework to constrain the number of degrees of freedom of a deforming object, we can model more complex types of deformation and demonstrate success in cases where existing techniques fail.

The proposed method first relies on a new initialization that quantizes the non-rigidity into a temporal succession of rigid shapes. An optimization then follows to minimize, at the same time, the reprojection error as well as constraints on the smoothness of the 3D shape deformations. A constraint on the shape manifold dimensionality is also enforced to make sure the recovered 3D shapes have a given number of degrees of freedom.

4.1 Problem

We want to recover the different parameters (camera ones $(R_t, \mathbf{t}_t)$ (2×3 and 2×1 matrices respectively as defined earlier) and shape ones S_t) and they are a solution of the following *reprojection error* minimization:

$$\arg \min_{R_t, \mathbf{t}_t, S_t} \sum_{t=1}^{f} \left\| \left(R_t S_t + \mathbf{t}_t \mathbf{1}_n{}^\top \right) - W_t \right\|_F^2 \tag{4.1}$$

As this equation is severely underconstrained, some assumptions need to be made.

Camera Motion

First, we assume the camera motion is smooth: the camera position does not change much from one frame to the next. Therefore, we add two regularization terms to Equation (4.1):

$$\lambda_R \sum_{t=2}^{f} \| R_t - R_{t-1} \|_F^2 + \lambda_{\mathbf{t}} \sum_{t=2}^{f} \| \mathbf{t}_t - \mathbf{t}_{t-1} \|_F^2 \tag{4.2}$$

where λ_R and $\lambda_{\mathbf{t}}$ are regularization constants.

Smooth Deformation

Next, another valid assumption is that the observed object does not change much from one frame to the next: this is also a physical constraint that is usually assumed by the feature tracker prior to SFM. Hence a new term is added to Equation (4.1):

$$\lambda_S \sum_{t=2}^{f} \| S_t - S_{t-1} \|_2^2 \tag{4.3}$$

where λ_S is another regularization constant. Higher order smoothness terms could be used, but this one proved sufficient for all our experiments.

Degrees of Freedom

While previous methods only allow for linear deformations, the proposed approach only constrains the shape deformation to have at most d *degrees of freedom*, hence allowing for non-linear deformations. This means that all the possible shapes S_t lie on a d-dimensional *manifold* or that, locally, several nearby shapes lie on a d-dimensional linear subspace. We will make this constraint explicit in Section 4.2.2.

Complete Problem Formulation

Our problem can now be re-formulated as the following optimization:

$$\min_{R_t,\mathbf{t}_t,S_t} \sum_{t=0}^{f} \left\| \left(R_t S_t + \mathbf{t}_t \mathbf{1}_n{}^\top \right) - W_t \right\|_F^2 + \lambda_S \sum_{t=2}^{f} \| S_t - S_{t-1} \|_F^2 \\ + \lambda_R \sum_{t=2}^{f} \| R_t - R_{t-1} \|_F^2 + \lambda_{\mathbf{t}} \sum_{t=2}^{f} \| \mathbf{t}_t - \mathbf{t}_{t-1} \|_F^2 \quad (4.4)$$

with the S_t's constrained to lie on a d-dimensional manifold.

Ambiguities

In SFM problems, there is usually an overall rigid ambiguity on the camera position. In our formulation, if the 3D shapes are modified by an overall rigid transform $(R, \mathbf{t})$, we obtain the following new unknowns: $S'_t = RS_t + \mathbf{t}$, $R'_t = R_t R^\top$ and $\mathbf{t}'_t = \mathbf{t}_t - R'_t \mathbf{t}$. As $R_t S_t + T_t = R'_t S'_t + T'_t$, the first term of Equation (4.4) will not change. As the Frobenius norm is rotation-invariant, the terms 2 and 3 will also be unchanged. Nonetheless, the last term becomes: $\left\| \mathbf{t}'_t - \mathbf{t}'_{t-1} \right\|_2 = \left\| \mathbf{t}_t - \mathbf{t}_{t-1} - (R'_t - R'_{t-1})\mathbf{t} \right\|_2 \neq \| \mathbf{t}_t - \mathbf{t}_{t-1} \|_2$ except if $\mathbf{t} = \mathbf{0}$.

Therefore, with our formulation, there is only a global *rotation ambiguity*, that we resolve by imposing $R_1 = \mathrm{I}_3$.

Also, the third component of the $\mathbf{t}_t$'s only matters in the last term of Equation (4.4) and a trivial optimum is reached by setting them to the same value. This arbitrary value is an ambiguity inherent to the orthographic model.

Over-Constrained Problem

We must recover $3f$ camera rotation angles, $2f$ camera translation parameters (2 and not 3, as there is a depth ambiguity with orthographic cameras) and $3n \times f$ 3D shape parameters. On the other hand, $2n \times f$ coordinates are given in the W_t matrices.

A d-dimensional linear subspace is parametrizable by a point and a basis of d elements, each of size $3n$. The point can be chosen anywhere in the subspace and therefore has $3n - d$ degrees of freedom. Also, the basis only has $3nd - d^2$ degrees of freedom. The subspace can therefore be explained with $3n - d + 3nd - d^2 = (d + 1)(3n - d)$ parameters.

In our formulation, the shapes lie on a d-dimensional manifold, and locally on a d-dimensional linear subspace. Therefore, if the data is uniformly distributed on the manifold, there exists a neighborhood size s such that every neighborhood of s shapes approximately lies on a d-dimensional *linear* subspace. Consequently, every s-neighborhood can be explained by s linear combinations and a d-dimensional subspace. As a result, each frame can be explained, on average, by $\frac{1}{s}(sd + (d + 1)(3n - d)) = d + \frac{1}{s}(d + 1)(3n - d)$ parameters.

As shown in Table 4.1, our model requires more parameters than traditional SFM techniques but it is still over-constrained provided $5f + fd + \frac{f}{s}(d + 1)(3n - d) < 2nf$ or again:

$$5 + d + \frac{1}{s}(d + 1)(3n - d) < 2n \tag{4.5}$$

To give a sense of magnitude, we can assume that usually, $d < 10$ and $n > 100$. Therefore, the inequality can be approximated by $s \gtrsim 1.5(d + 1)$. Practically, this implies that the observed shapes have to appear in *similar* configurations at least $1.5(d + 1)$ times (*similar* but not necessarily *exact*: our approach does not require a perfect repetitiveness of the motion).

Table 4.1: Number of parameters defining the model in rigid SFM, traditional Non-Rigid SFM, and the proposed approach (all in the orthographic case). This statistics concerns a sequence of f frames with n features whose shape lie on a d-dimensional subspace ($d = 0$ in the rigid case). s is such that every s neighboring shapes constitute a linear subspace.

	Rigid	Classical NRSFM	Proposed NRSFM
camera	$3f + 2f$		
basis	—	$(d + 1)(3n - d)$	—
shape	$3n$	d	$d + \frac{1}{s}(d + 1)(3n - d)$
total	$5f+3n$	$5f+fd+(d+1)(3n-d)$	$5f+fd+\frac{f}{s}(d+1)(3n-d)$

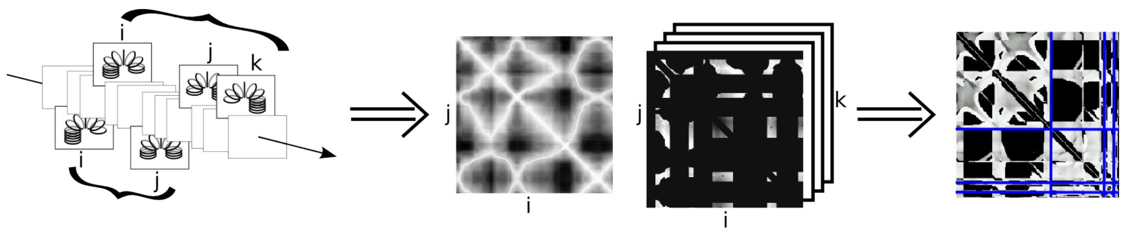


Figure 4.1: NRSFM is initialized by a *Rigid Shape Chain*. First of all, pairs and triplets of frames of the original video sequence are extracted. Each pair or triplet is then assumed to be projections of the same 3D view: the corresponding *reprojection error* is computed and used to define an affinity matrix/affinity tensor (only triplet composed of pairs with high affinities are considered for efficiency, hence the missing data in the image). These are then combined into a hypergraph that is flattened and clustered using *clique expansion* and *normalized cut*. The resulting clusters represent prototypical 3D shapes and they are then aligned to create a first estimate of the S_t's.

4.2 Method

4.2.1 Initialization

In order to minimize Equation (4.4), several techniques will be used including a partial closed-form solution, gradient descent and manifold denoising. The system is initialized by assuming that the object can be modeled at any frame as a rigid transformation of one of a collection of shape templates.

Hypergraph Interpretation

We first cluster the shapes S_t. We assume that if S_t and $S_{t'}$ are in the same cluster, they are derived from different rigid transformations of the same shape template and the corresponding reprojection error $\text{err}_{tt'}$ is computed. We can then interpret the video sequence as a graph whose nodes are the frames and whose edges are weighted with the following reprojection affinity:

$$w_{tt'} = e^{-\frac{\text{err}_{tt'}^2}{\sigma^2}}$$

An affinity close to 0 means the two 3D shapes are not in the same state. On the other hand, if it close to 1, it can indicate either a similar state, or an ambiguity due to a view point that "hides" the shape differences.

In order to disambiguate these cases, we compute higher order affinities using the reprojection error of triplets of frames using Tomasi and Kanade (1992). This method is known to be more stable than epipolar geometry but is also more expensive. Therefore, only pairs of frames that already have a high pairwise epipolar-based affinity are considered to form triplets.

Once these dyadic and triadic affinities have been computed, we obtain a hypergraph with dyadic and triadic connections. In order to cluster the object shapes, the hypergraph is approximated by a graph using *clique expansion* Schweikert and Kernighan (1972).

Rigid Shape Chain

In order to deal with noise and outliers and to lower the dimensionality of the initialization, *quantization* is performed on the frames by separating them into clusters of equivalent 3D shape. To this end, a simple K-way clustering is applied using *normalized cuts* Shi and Malik (2000) to the previously defined frame affinity graph. K is chosen as the biggest number of clusters such that no cluster has less than 3 frames (in practice, K was between 10 and 30 for a 200-frame sequence).

Next, each cluster $\mathbf{C}^i$ is considered and its corresponding 3D shape $\mathbf{S}^i, i = 1, \ldots, K$ is computed (using Tomasi and Kanade (1992)).

The resulting clustering is an initialization that actually explains the observed non-rigidity by a succession of rigid problems. We name this approach a *Rigid Shape Chain* (*cf*.Figure 4.1).

Initial Shape Alignment

The shapes obtained so far have been computed without regard to any deformation smoothness. This smoothness is enforced by applying a rigid transform $(\mathbf{R}^i, \mathbf{T}^i)$ to the $\mathbf{S}^i$'s in order to minimize the following temporal smoothness criterion:

$$\min_{(\mathbf{R}^i, \mathbf{T}^i)} \sum_{t=2}^{f} \left\| \mathbf{R}^{i(t)} \mathbf{S}^{i(t)} + \mathbf{T}^{i(t)} - \mathbf{R}^{i(t-1)} \mathbf{S}^{i(t-1)} - \mathbf{T}^{i(t-1)} \right\|_F^2 \tag{4.6}$$

where $i : t \mapsto i | S_t \in \mathbf{C}^i$. This minimization is simply a continuous version of the *exterior orientation* problem Horn (1986). It can be solved by least squares optimization with random initialization and rotation constraints on the $\mathbf{R}^i$'s. In practice, we found that finding the closed form solution of the problem with no rotation constraints, and then projecting it onto $SO(3)$ Horn (1987) gave very good and fast results. The advantage of this approach is that it can rectify 3D shapes that have been flipped because of the possible *chirality* ambiguities appearing during the rigid shape chain.

Finally, the $\mathbf{R}^{i(t)} \mathbf{S}^{i(t)} + \mathbf{T}^{i(t)}$ are set to be the initial estimates of the S_t's (modulo a global rotation to ensure that $R_1 = \mathrm{I}_3$).

4.2.2 Minimization

After initialization, we obtain a reasonable approximation of the shapes and camera positions over time. The minimization of Equation (4.4) proceeds by alternating between the different unknowns, assuming the others are fixed.

Optimizing Camera Positions

If the S_t's and R_t's are fixed, finding a global optimum to Equation (4.4) with respect to $\mathbf{t}$ is trivial: it is the closed form solution of a sparse linear system.

Concerning the R_t's, the global optimum is as trivial, provided they are not constrained to be rotation matrices. In practice, we compute this global optimum and project it to $SO(3)$. If the result lowers the error, it is kept. Otherwise, we perform a projection-based gradient descent (at every step of gradient descent, the result is reprojected onto $SO(3)$).

LSML

When performing optimization on the S_t's, there are two criteria to take into account: the smoothness term in Equation (4.4) and the shape manifold dimensionality constraint. Optimizing the first one is just a least square optimization, but the second one needs a new interpretation.

The problem of imposing this dimensionality constraint can be seen as trying to force the S_t to lie on a d-dimensional manifold (as previously defined, d is the number of freedom of the observed object). After initialization, the S_t's are close to this manifold but are not on it: it is as if they formed a noisy low-dimensional manifold. As mentioned earlier, we have no reason to believe that this manifold is planar or isometric to a plane, hence our motivation for using LSML Dollár et al. (2007).

LSML is a manifold learning technique that seeks to learn from training data a smooth mapping from every point on the manifold to its local tangents. Consider:

$$\mathcal{M} : \begin{array}{l} \mathbb{R}^d \to \mathbb{R}^D \\ \mathbf{y} \mapsto \mathbf{x} \end{array} \tag{4.7}$$

a smooth mapping from a low d-dimensional space to a higher D-dimensional space (*e.g.* $\mathbf{y}$ is the coordinate on a manifold of a high dimensional point x), $d \ll D$. LSML seeks to recover the mapping:

$$\mathcal{H} : \begin{array}{l} \mathbb{R}^D \to \mathbb{R}^{D \times d} \\ \mathbf{x} \mapsto \left[\partial/\partial \mathbf{y}_1 \mathcal{M}(\mathbf{y}) \dots \partial/\partial \mathbf{y}_d \mathcal{M}(\mathbf{y})\right] \end{array} \tag{4.8}$$

where $\mathcal{M}(\mathbf{y}) = \mathbf{x}$ and $\mathbf{y} = \left[\mathbf{y}_1 \dots \mathbf{y}_d\right]^\top \in \mathbb{R}^d$.

The strength of this technique is that the mapping is not only learned for the training points but, by continuity, it is applicable to any new given point in $\mathbb{R}^D$.

LSML can also learn $\mathcal{H}$ from noisy data and then denoise it by making the points follow the gradient of an optimization criterion detailed in Dollár et al. (2007). It is important to notice that LSML is limited to noise orthogonal to the manifold and cannot deal with collinear noise.

At each of our optimization steps, LSML is used to learn the manifold of noisy S_t's and recover the gradient for the LSML noise criterion.

Optimizing 3D Shapes

Optimizing the S_t's now needs to take two criteria into account - the one from Equation (4.4) and the one from LSML- and we have computed an optimization gradient for both, which we define as ∇_{Smooth} and ∇_{LSML}. It is an instance of *multi-objective optimization*. As, we do not want one of the constraints to be enforced more and impede the other, we decide not to use a weighted linear combination of the two criteria or a Lagrangian multiplier: we keep the constraints separate and and optimize them at the same time using *multi-level programming* to favor the dimensionality contraint.

Instead of using our two gradients as they are, we keep ∇_{LSML} responsible for any variation orthogonal to the shape manifold but only restrict ∇_{Smooth} to its projection

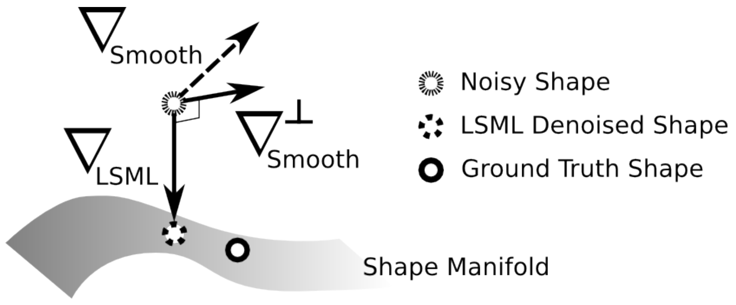


Figure 4.2: Two gradients are involved when optimizing the S_t's. First, there is a gradient ∇_{LSML} provided by LSML that tends to bring a noisy S_t back onto the shape manifold (but that is only orthogonal to it). Then, there is a gradient ∇_{Smooth} that minimizes the smoothness of the shape deformation. In order not to interfere with the LSML gradient, only its component orthogonal to ∇_{LSML} is considered: $\nabla^{\perp}_{\text{Smooth}}$. A linear combination of these two gradients is then searched to optimize the two criteria at the same time.

$\nabla^{\perp}_{\text{Smooth}}$ onto a plane tangent to ∇_{LSML}: this way, $\nabla^{\perp}_{\text{Smooth}}$ does not interfere with any effect of ∇_{LSML}. Figure 4.2 illustrates this approach.

What follows is a gradient descent step following the gradient: $\nabla = a\nabla_{\text{LSML}} + b\nabla^{\perp}_{\text{Smooth}}$, where a and b are chosen so that both criteria are optimized at the same time.

Outliers

As LSML is not robust to outliers, special care is taken for any 3D shape that does not comply to the two following criteria:

- the distance to one of its neighbors is above three standard deviations (of the distribution of distances from points to their neighbors)
- the distance to its temporal predecessor is above three standard deviations.

These points are simply optimized by disregarding the manifold dimensionality constraint and by assigning them to their globally optimal value (which can be obtained in closed form, in a similar way as the $\mathbf{t}_t$'s).

Considerations

Each iteration of our optimization routine attains a lower error for Equation (4.4) that in the previous one, so it is bound to converge. In our experiments, we did not need to repeat the optimization (which could have been useful as it contains randomized algorithms such as Normalized Cut or LSML) but we faced a slow convergence (100 to 200 iterations were required).

Also, most of the steps described previously take a few seconds to compute except for the triadic affinity computation and the S_t gradient descent involving LSML. Indeed, for each iteration, LSML needs to be retrained which can take up to a few minutes leading to an overall time of an hour or two.

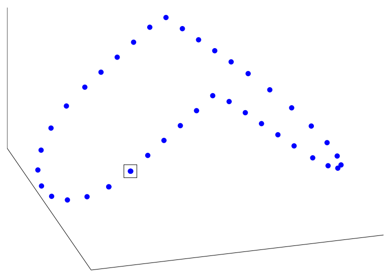
4.3 Experiments

We experimented our method on both synthetic and real data. These experiments show the flexibility of our approach and its robustness as well as comparisons with state of the art using code from Torresani et al. (2008) and our own implementation of Xiao and Kanade (2005). We will respectively refer to these two methods as THB and XCK while we name ours *Manifold Structure From Motion* or MSFM.

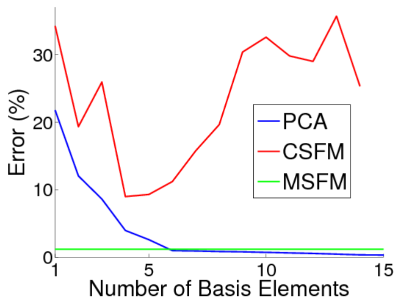
4.3.1 Synthetic Data

Roller Coaster

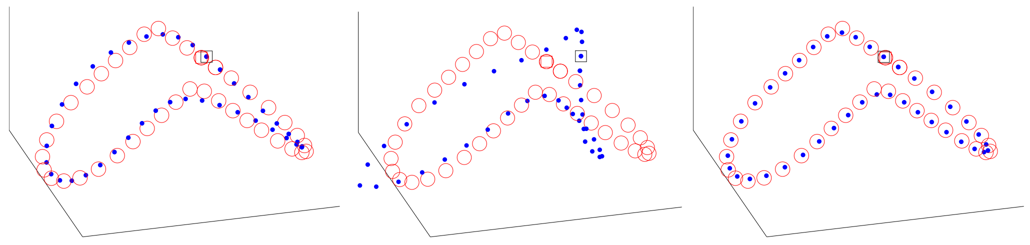
The first data set is a synthetic roller coaster. The video sequence consists of 200 frames with 42 points moving on a fixed closed track. As seen on Figure 4.3(a), it looks like a closed roller coaster or a bent bike chain. The camera rotates around the object while it deforms. Both the object and the camera evolve at random speeds (no translation is involved for the camera).



(a) Frame Example



(b) Reconstruction Performance



(c) Reconstruction Examples (respectively PCA, THB, MSFM)

Figure 4.3: **Roller Coaster.** Figure 4.3(a) is a typical frame from the sequence. The black square refers to the first point of the coaster. Overall, the coaster loops 6 times. Figure 4.3(b) illustrates the performances of the different algorithms the percentage error refers to the average reconstruction error along the camera depth axis, normalized by the depth of the roller coaster as in Torresani et al. (2008)). PCA on the 3D points beats MSFM when the basis contains at least 6 elements which shows that the structure is not easily representable by a linear basis. Figure 4.3(c) shows examples of reconstruction for PCA (with 5 shapes in the basis), THB (with 4 shapes in the basis) and MSFM. The corner of the roller coaster presented a challenge for PCA to capture even with a bigger basis.

This motion only has one degree of freedom as the points have to move along a fixed structure. Nonetheless, this motion causes problem for THB because it is not easily representable by a linear basis (for comparison, we show how hard it is for PCA to characterize the data given the full 3D points in Figure 4.3(b)). Also, the object does not have a main component that could be considered as rigid and be used as initialization for THB.

THB seems to fail in this case while MSFM only has 1.2% of error (the computed error is similar to Torresani et al. (2008) and detailed in Figure 4.3). Figure 4.3(b) also shows an interesting limitation of using a linear basis: as its focus is to minimize the reprojection error at any cost, more elements in the basis can help lowering it but at the cost of getting a worse 3D reconstruction.

It is also worth mentioning that the sequence of recovered S_t is also moving on itself (in addition to its intrinsic one degree of freedom): this is due to the fact that this optimizes the overall smoothness.

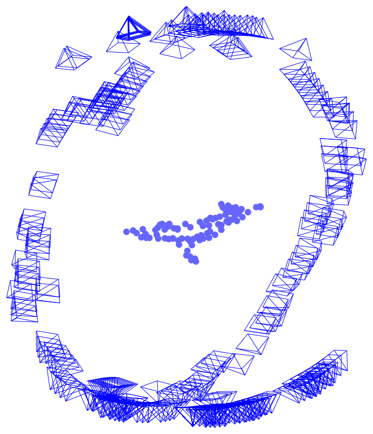
Bending Shark

The next experiment uses the shark data from Torresani et al. (2008). It consists of 240 frames during which 91 points form a shark that bends its tail left/right or up/down (hence 2 degrees of freedom). In Torresani et al. (2003), they obtain errors of 1.24% and 2.5%. With our setup, we obtain 3%, which is comparable to their second best method. Several details are shown in Figure 4.4

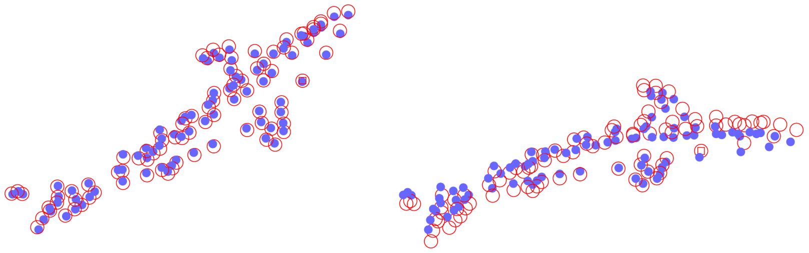
4.3.2 Real Data

The final round of experiments involves a calibrated video sequence of a Slinky® toy: this spring has a complex motion but, in this instance, only one degree of freedom. 27 painted features were tracked during a 300 frame long video sequence. There are approximately three periods of up/down movement that occur.

This data set is difficult for two reasons: the feature trajectories are noisy and the baseline between two extreme views is small. MSFM reconstructed a 1D-manifold of the different 3D-shapes with an average reprojection error of 1.4%. Examples are illustrated in Figure 4.5. THB fails in this case as the object does not have a main rigid part and because it undergoes non-linear deformations (like compression).

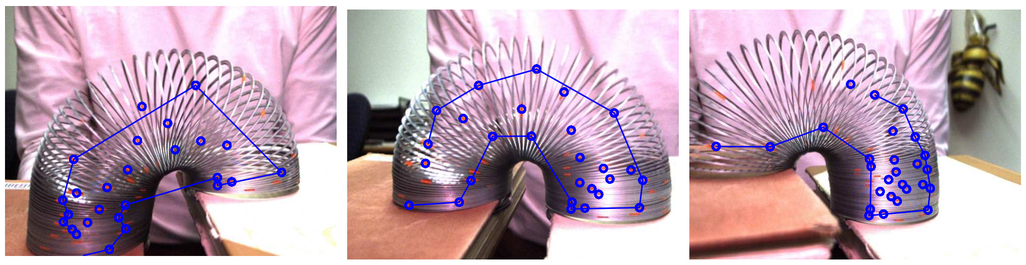


(a) MSFM Initialization

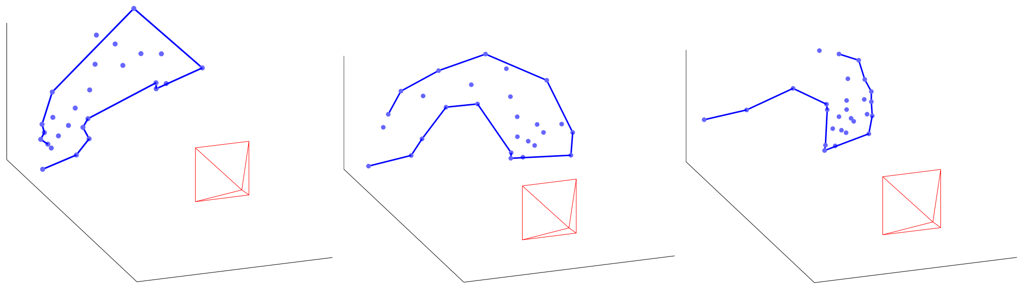


(b) New Views

Figure 4.4: **Bending Shark.** Figure 4.4(a) illustrates the initial camera position estimation after the *rigid shape chain* computation. The ground truth camera movement of the camera is a view from below first followed by a full rotation around the shark. There are of course a few outliers but the overall camera trajectory is already well approximated. Figure 4.4(b) shows two reconstructions of the shark sequence. In the right image, the camera was pointing down leading to a lower quality reconstruction because of the depth ambiguity.



(a) Some Frames of the Slinky Sequence



(b) MSFM Reconstruction

Figure 4.5: **Slinky.** Figure 4.5(a) shows a few frames of the slinky sequence with some tracked features. The lines are just drawn to help visualize the 3D structure in the reconstruction Figure 4.5(b). These three frames demonstrate the compression the object undergoes: this property is difficult for a linear basis to model, even in 2D, hence the failure of THB. On the other hand, MSFM seems to recover correct 3D feature positions: the structure contains compression and seems to have a correct orientation.

4.4 Acknowledgements

Chapter 4, is a reprint of the material as it appears in Re-Thinking Non-Rigid Structure from Motion, V. Rabaud and S. Belongie, IEEE Conference on Computer Vision and Pattern Recognition, 2008.

Chapter 5

Conclusion

5.1 Contributions

Throughout this dissertation, we have developed and built upon a new interpretation of non-rigid structure from motion. The manifold of all the possible shapes of a 3D object is smooth and has a dimensionality that reflects its number of degrees of freedom, which is usually low. By focusing on studying this object first instead of having a matrix interpretation, we showed how we can reach performance better than the current state of the art techniques.

By keeping the linear manifold assumption, we showed how our approach TSFM performs better, even in noisy cases. As TSFM relies on recent math development we can expect more robust and faster performance in the upcoming years. Mostly, SDP problems with rank constraints have been the subject of recent research, and improvements in the current techniques could drastically improve our approach.

Finally, this thesis was also the occasion to make a theoretical jump and move to looser assumptions concerning the shape manifold by only restricting its dimensionality. Here again, we showed how MSFM can overcome problems with current techniques and assumptions by being more flexible.

5.2 Future Work

There are a few simple extensions that could be added to TSFM and MSFM. First, just like in previous SFM works, dealing with occlusions and the projective case could be easily incorporated. This is because only pairs and triplets of frames are considered, and the theory for such simple cases has been established a decade or more ago.

Also, it might be possible to incorporate TSFM into MSFM just to obtain the rigid shape chain. As only clusters of coherent frames need to be used, just the manifold recovery part of TSFM could be used to this end.

It seems that more information about the object could be used when recovering the embedding. Information like periodicity. exact state repetitions, object symmetry, and deformation curvature on the manifold could help constrain the shape manifold even more and probably help dealing with the sampling issues by providing stronger constraints.

Finally, and this is more of a dream at this stage: it should also be possible to move away from the feature paradigm and incorporate dense information into the pipeline: in TSFM, the shape manifold only involves coordinates in a shape basis (or local shape basis as in MSFM). These have no tie with the basis itself, which could help abstracting it to some dense basis. But sometimes dreams do come true ...

Appendix A

General Math Considerations

A.1 Notations

Table A.1: The different math notations used throughout this thesis are explained in the table below.

Variables	Definition
a	Scalar
$\mathbf{a}$	Vector
$\mathbf{a}^i$	Vector indexed for some purpose
a_i	i^{th} element of the vector $\mathbf{a}$
A	Matrix
A^i	Matrix indexed for some purpose
$A_{\cdot j}$	j^{th} column of the matrix A
$A_{i\cdot}$	i^{th} row of the matrix A
$A_{i:j,k:l}$	submatrix of the matrix A composed of the rows from i to j and the columns from k to l.
$A_{\cdot,k:end}$	submatrix of the matrix A composed of the k^{th} column up to the last.

I, I_n	identity matrix, identity matrix of size $n \times n$.
$\mathbf{0}, \mathbf{0}_n$	vector composed of 0's only. If its dimensionality n is ambiguous, it is indicated as a lower index.
$0, 0_n$	matrix composed of 0's only. If it is square and its dimensionality n is ambiguous, it is indicated as a lower index.
$\mathbf{1}, \mathbf{1}_n$	same as the **0** case defined above.
$1, 1_n$	same as the 0 case defined above.
$\text{vec}\,(A)$	vector version of the matrix A
$A \otimes B$	Kronecker product of A by B
A^+	Pseudo-inverse of A
$\\|\mathbf{x}\\|_2^2$	squared L^2 norm of $\mathbf{x}$
$\\|A\\|_F^2$	squared Frobenius norm of A
$[\![i; j]\!]$	set of integers between i and j
$\mathcal{C}_n$	$n \times (n+1)$ matrix such that $\mathcal{C}_n = \begin{bmatrix} 1 & 0 & \dots & 0 & -1 \\ 0 & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & -1 \end{bmatrix}$

A.2 General Linear Algebra

A.2.1 General Rules

If A, B, X are matrices:

$$\text{vec}\,(AXB) = (B^\top \otimes A)\text{vec}\,(X) \tag{A.1}$$

$$\text{tr}\,(AB) = \text{vec}\,(A)^\top \text{vec}\,(B) = \text{tr}\left(B^\top A^\top\right) = \text{vec}\left(A^\top\right)^\top \text{vec}\left(B^\top\right) \tag{A.2}$$

$$(\mathrm{I}_n + 1_n)^{-1} = \mathrm{I}_n - \frac{1}{n+1} 1_n \tag{A.3}$$

The solution of $\min \|AX - B\|_F^2$ is $X = A^+B + V$ where V is such that $AV = 0$. Equivalently, the solution of $\min \|XA - B\|_F^2$ is of the form: $X = BA^+ + V$ where

Table A.2: The different notations related to structure from motion are detailed in the table below.

Variables	**Definition**
n	Number of tracked features in the video sequence
f	Number of frames in the video sequence
s	Number of basis element in the shape basis
W_t	$2 \times n$ *measurement matrix*: each column contains the measured coordinates of a feature on frame t
W_t^*	$3 \times n$ *measurement matrix*: each column contains the measured 3D-coordinates of a feature on frame t. This quantity is never measured, but is used during computation. Only a projection of these measured 3D coordinates is give via W_t
S_t	$3 \times n$ *shape matrix*: each column contains the 3D coordinates of a feature at time t
$(R_t^*, \mathbf{t}_t^*)$	3×3 rotation matrix and 3×1 translation matrix at time t defining the camera position
$(R_t, \mathbf{t}_t)$	R_t and $\mathbf{t}_t$ contains the first two rows of R_t^* and $\mathbf{t}_t^*$

VA=0.

A.2.2 The Curious Case of $\mathcal{C}_n$

$\mathcal{C}_n$ is defined in the previous tables and by definition: $\mathcal{C}_n\mathcal{C}_n^\top = \mathbf{I}_n + \mathbf{1}_n$.

A.3 Derivations in Linear Algebra

If A, B, C, D, X are matrices:

$$\partial\|A\|_F^2 = \partial\text{tr}\left(AA^\top\right) = \text{tr}\left(\partial AA^\top + A(\partial A)^\top\right) = 2\text{tr}\left(A(\partial A)^\top\right) \tag{A.4}$$

Now, if $A = BXC + D$, and we take the derivative with respect to X:

$$\frac{\mathrm{d}}{\mathrm{d}X}\|A\|_F^2 = \frac{\mathrm{d}}{\mathrm{d}X}\text{tr}\left((BXC+D)(BXC+D)^\top\right) \tag{A.5}$$

$$= \frac{\mathrm{d}}{\mathrm{d}X}\text{tr}\left(BXC(BXC)^\top\right) + 2\text{tr}\left(BXCD^\top\right) \tag{A.6}$$

$$= B^\top BXCC^\top + B^\top BXCC^\top + 2B^\top DC^\top \tag{A.7}$$

$$= 2B^\top(BXC+D)C^\top \tag{A.8}$$

Chain Rule: if $U = f(X)$, then:

$$\frac{\partial}{\partial X_{ij}}g(U) = \text{tr}\left(\left(\frac{\partial}{\partial U}g(U)\right)^\top \frac{\partial}{\partial X_{ij}}U\right) \tag{A.9}$$

Appendix B

Linear Embedding Formulas

B.1 Infimum Computation

We want to solve for:

$$\begin{aligned} \min_{R^*,\mathbf{x},\mathbf{y}} \quad & \left\| \begin{bmatrix} W_1 \\ \mathbf{x}^\top \end{bmatrix} - R^* \begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} \right\|_F^2 \\ \text{subject to} \quad & R^* \text{ is a rotation matrix} \\ & \text{centering conditions: } \overline{\mathbf{x}} = 0, \overline{\mathbf{y}} = 0 \end{aligned} \tag{B.1}$$

$\mathbf{x}$ can be chosen to nullify the third row of the difference, leading to the following new problem:

$$\begin{aligned} \min_{R,\mathbf{y}} \quad & f(R,\mathbf{y}) = \left\| W_1 \mathcal{C}_{n-1} - R \begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} \mathcal{C}_{n-1} \right\|_F^2 \\ \text{subject to} \quad & R \text{ is the top two rows of a rotation matrix} \end{aligned} \tag{B.2}$$

with $\mathbf{y}, W_1$ and W_2 being shorter by one element, $\mathcal{C}_{n-1} = \begin{bmatrix} 1 & 0 & \dots & 0 & -1 \\ 0 & \ddots & & \vdots & \vdots \\ \vdots & & \ddots & 0 & \vdots \\ 0 & \dots & 0 & 1 & -1 \end{bmatrix}$ such that $\mathbf{y}^\top \mathcal{C}_{n-1} = \begin{bmatrix} \mathbf{y}^\top & -\mathbf{y}^\top \mathbf{1} \end{bmatrix}$ (centering condition is forced) and R parametrized by a quaternion $(a,b,c,d)^\top$.Hence:

$$\frac{\partial}{\partial \mathbf{y}_i} f(R,y) = \text{tr}\left(\frac{\partial}{\partial B}\left(\|W_1\mathcal{C}_{n-1} - RB\mathcal{C}_{n-1}\|_F^2\right)^\top \frac{\partial}{\partial \mathbf{y}_i}\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix}\right) \tag{B.3}$$

$$= \text{tr}\left(2\mathcal{C}_{n-1}\left(RB\mathcal{C}_{n-1} - W_1\mathcal{C}_{n-1}\right)^\top R\frac{\partial}{\partial \mathbf{y}_i}\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix}\right) \tag{B.4}$$

$$= \text{tr}\left(2\mathcal{C}_{n-1}\mathcal{C}_{n-1}^\top\left(R\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} - W_1\right)^\top R\begin{bmatrix} & & & 0 & & & \\ 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}\right) \tag{B.5}$$

$$= \text{tr}\left(2(\mathbf{I}_{n-1} + \mathbf{1}_{n-1})\left(R\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} - W_1\right)^\top R_{\cdot 3}\begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix}\right) \tag{B.6}$$

$$= \begin{bmatrix} 0 & \dots & 0 & 1 & 0 & \dots & 0 \end{bmatrix} 2(\mathbf{I}_{n-1} + \mathbf{1}_{n-1})\left(R\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} - W_1\right)^\top R_{\cdot 3} \tag{B.7}$$

$$\frac{\partial}{\partial \mathbf{y}} f(R,y) = 2(\mathbf{I}_{n-1} + \mathbf{1}_{n-1})\left(R\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} - W_1\right)^\top R_{\cdot 3} \tag{B.8}$$

As it is null at the optimum and as $(\mathbf{I}_{n-1} + \mathbf{1}_{n-1})^{-1} = \mathbf{I}_{n-1} - \frac{1}{n}\mathbf{1}_{n-1}$ (and is therefore invertible), we have:

$$\begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix}^\top R^\top R_{\cdot 3} = W_1^\top R_{\cdot 3} \tag{B.9}$$

$$\begin{bmatrix} W_2^\top & \mathbf{y} \end{bmatrix} R^\top R_{\cdot 3} = W_1^\top R_{\cdot 3} \tag{B.10}$$

$$W_2^\top R_{\cdot,1:2}^\top R_{\cdot 3} + \mathbf{y} R_{\cdot 3}^\top R_{\cdot 3} = W_1^\top R_{\cdot 3} \tag{B.11}$$

$$R_{\cdot 3}^\top R_{\cdot 3}\mathbf{y} = W_1^\top R_{\cdot 3} - W_2^\top R_{\cdot,1:2}^\top R_{\cdot 3} \tag{B.12}$$

Therefore:

$$\mathbf{y} = \frac{1}{R_{\cdot 3}^\top R_{\cdot 3}} \left(W_1^\top - W_2^\top R_{\cdot,1:2}^\top\right) R_{\cdot 3} \tag{B.13}$$

Now:

$$f(R) = \left\| R \begin{bmatrix} W_2 \\ \mathbf{y}^\top \end{bmatrix} \mathcal{C}_{n-1} - W_1 \mathcal{C}_{n-1} \right\|_F^2 \tag{B.14}$$

$$= \left\| R_{\cdot,1:2} W_2 \mathcal{C}_{n-1} - W_1 \mathcal{C}_{n-1} + R_{\cdot 3} \mathbf{y}^\top \mathcal{C}_{n-1} \right\|_F^2 \tag{B.15}$$

$$= \left\| R_{\cdot,1:2} W_2 \mathcal{C}_{n-1} - W_1 \mathcal{C}_{n-1} + \frac{1}{R_{\cdot 3}^\top R_{\cdot 3}} R_{\cdot 3} R_{\cdot 3}^\top \left(W_1 - R_{\cdot,1:2} W_2\right) \mathcal{C}_{n-1} \right\|_F^2 \tag{B.16}$$

$$= \left\| \left(\frac{1}{R_{\cdot 3}^\top R_{\cdot 3}} R_{\cdot 3} R_{\cdot 3}^\top - \mathbf{I} \right) \left(W_1 - R_{\cdot,1:2} W_2\right) \mathcal{C}_{n-1} \right\|_F^2 \tag{B.17}$$

B.2 Computational Considerations

The following is useful to perform gradient descent on R. We pose:

$$W_1' = W_1 \mathcal{C}_{n-1}, \quad W_2' = W_2 \mathcal{C}_{n-1}, \quad W' = \begin{bmatrix} W_1' \\ W_2' \end{bmatrix} \tag{B.18}$$

$$R_1 = \frac{1}{R_{\cdot 3}^\top R_{\cdot 3}} R_{\cdot 3} R_{\cdot 3}^\top - \mathbf{I}, \quad R_2 = -R_1 R_{\cdot,1:2} \tag{B.19}$$

$$R' = \begin{bmatrix} R_1 & R_2 \end{bmatrix} \tag{B.20}$$

Then:

$$f(R) = \|R' W'\|_F^2 \tag{B.21}$$

$$\frac{\partial}{\partial a,b,c,d} f(R) = \mathrm{tr}\left(2 \left(R'W'\right)^\top \frac{\partial}{\partial a,b,c,d} \left(R'W'\right) \right) \tag{B.22}$$

$$= \mathrm{tr}\left(2 \left(R'W'\right)^\top \frac{\partial}{\partial a,b,c,d} (R') W' \right) \tag{B.23}$$

$$\frac{\partial^2}{\partial^2 a,b,c,d} f(R) = \mathrm{tr}\left(2 \left(R'W'\right)^\top \frac{\partial^2}{\partial^2 a,b,c,d} (R') W' \right) + \tag{B.24}$$

$$\mathrm{tr}\left(2 \left(\frac{\partial}{\partial a,b,c,d} (R') W' \right)^\top \left(\frac{\partial}{\partial a,b,c,d} (R') W' \right) \right) \tag{B.25}$$

Appendix C

LSML Appendix

C.1 Notations

C.2 Centered Error Function - General Form

The centered form of the error function is:

$$\text{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta \left(\frac{\mathbf{x}^i + \mathbf{x}^j}{2} \right) \boldsymbol{\epsilon}^{ij} - (\mathbf{x}^j - \mathbf{x}^i) \right\|_2^2 \tag{C.1}$$

$$= \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j \in \mathcal{N}^i} \left\| \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) \boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j} \right\|_2^2 \tag{C.2}$$

Here $\overline{\mathbf{x}}^{ij} = \frac{\mathbf{x}^i + \mathbf{x}^j}{2}$ and $\Delta^i_{\cdot j} = \mathbf{x}^j - \mathbf{x}^i$. If we treat $\Delta^i_{\cdot j}$ as an approximation of a directional derivative on the manifold, then the approximation is best for the point on the manifold directly between $\mathbf{x}^i$ and $\mathbf{x}^j$. To see that this is the case, consider that given a function f, the centered approximation of the derivative $\frac{\partial f}{\partial x} = (f(x+h) - f(x-h))/2h$ has no second order error while the un-centered approximation $\frac{\partial f}{\partial x} = (f(x+h) - f(x))/h$ does. If $\mathbf{x}^i$ and $\mathbf{x}^j$ are close enough so that a linear approximation of the manifold is sufficient then the un-centered form works well. If, however, a quadratic approximation is necessary, the centered version prevails.

We can explicitly enforce the mapping $\mathcal{H}_\theta$ to be smooth by adding a regularization term to the error function (in addition to implicit smoothness that may come from

Table C.1: Table describing the different variables used in the LSML algorithm

Variables	**Dimensions**	**Definition**
D		dim. of original space
d		dim. of projected space
n		number of data points
f		number features per point
$\mathbf{x}^i$	$[D \times 1]$	$i \in [n]$, data point
$\overline{\mathbf{x}}^{ij}$	$[D \times 1]$	$\overline{\mathbf{x}}^{ij} = \frac{\mathbf{x}^i + \mathbf{x}^j}{2}$
$\mathbf{f}^{ij}$	$[f \times 1]$	features of $\overline{\mathbf{x}}^{ij}$
$\mathcal{N}^i$		indices of neighbors of $\mathbf{x}^i$
$\mathcal{H}_\theta$		$\mathcal{H}_\theta : \mathbb{R}^D \to \mathbb{R}^{D \times d}$
H^{ij}	$[D \times d]$	$H^{ij} = \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})$ (for θ fixed)
$\boldsymbol{\epsilon}^{ij}$	$[d \times 1]$	alignment free parameter
Δ^i	$[D \times \lvert\mathcal{N}_i\rvert]$	$\Delta^i_{\cdot j} = \mathbf{x}^j - \mathbf{x}^i$

the form of $\mathcal{H}_\theta$ itself). For each i, the learned tangents at two neighboring locations $\overline{\mathbf{x}}^{ij}$ and $\overline{\mathbf{x}}^{ij'}$ should be similar, *i.e.* $\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'})\|_F^2$ should be small. To ensure that the $\mathcal{H}_\theta$'s do not get very small and the $\boldsymbol{\epsilon}$'s very large, $\|\boldsymbol{\epsilon}^{ij}\|_2^2$ must also be constrained. The regularized error function is:

$$\begin{aligned}\mathrm{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \left\{ \sum_{i,j \in \mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j}\right\|_2^2 + \lambda_\epsilon \sum_{i,j \in \mathcal{N}^i} \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 \right\} \\ + \lambda_\Theta \sum_{i,(j,j') \in \mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'})\right\|_F^2 \end{aligned} \qquad \text{(C.3)}$$

where λ_Θ is the normalized λ_ϵ: $\lambda_\Theta = \lambda_\epsilon \frac{\sum \#\mathcal{N}^i}{\sum \frac{1}{2}\#\mathcal{N}^i(\#\mathcal{N}^i - 1)}$ It can be shown that using separate λs for the two regularization terms is equivalent to having a shared λ (set accordingly).

C.3 Centered Minimization for General Manifolds

The linear parametrization of $\mathcal{H}_\Theta$ is specified by $\Theta = (\Theta^1, \cdots, \Theta^D)$, where each Θ^k is a $d \times f$ matrix. The total number of parameters is thus Ddf. $\mathcal{H}_\Theta$ has the form: $\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) = \left[\Theta^1 \mathbf{f}^{ij} \cdots \Theta^D \mathbf{f}^{ij}\right]^\top$. The error function becomes:

$$\begin{aligned}
\operatorname{err}(\theta) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} &\left\{ \sum_{i,j\in\mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j}\right\|_2^2 + \lambda_\epsilon \sum_{i,j\in\mathcal{N}^i} \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 \right\} \\
&+ \lambda_\Theta \sum_{i,(j,j')\in\mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\mathbf{x}}^{ij}) - \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij'})\right\|_F^2 \qquad \text{(C.4)} \\
= \min_{\{\boldsymbol{\epsilon}^{ij}\}} &\left\{ \sum_{i,j\in\mathcal{N}^i} \sum_{k=1}^{D} \left(\mathbf{f}^{ij\top}\Theta^{k\top}\boldsymbol{\epsilon}^{ij} - \Delta^i_{kj}\right)^2 + \lambda_\epsilon \sum_{i,j\in\mathcal{N}^i} \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 \right\} \\
&+ \lambda_\Theta \sum_{i,(j,j')\in\mathcal{N}^i} \sum_{k=1}^{D} \left\|\mathbf{f}^{ij\top}\Theta^{k\top} - \mathbf{f}^{ij'\top}\Theta^{k\top}\right\|_F^2 \qquad \text{(C.5)} \\
= \min_{\{\boldsymbol{\epsilon}^{ij}\}} &\left\{ \sum_{i,j\in\mathcal{N}^i} \sum_{k=1}^{D} \left(\left(\boldsymbol{\epsilon}^{ij\top} \otimes \mathbf{f}^{ij\top}\right) \operatorname{vec}\left(\Theta^{k\top}\right) - \Delta^i_{kj}\right)^2 + \lambda_\epsilon \sum_{i,j\in\mathcal{N}^i} \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 \right\} \\
&+ \lambda_\Theta \sum_{k=1}^{D} \left\|\Delta_F \Theta^{k\top}\right\|_F^2 \qquad \text{(C.6)} \\
= \min_{\{\boldsymbol{\epsilon}^{ij}\}} &\left\{ \sum_{k=1}^{D} \left\|A\operatorname{vec}\left(\Theta^{k\top}\right) - \mathbf{b}^k\right\|_2^2 + \lambda_\epsilon \sum_{i,j\in\mathcal{N}^i} \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 \right\} + \lambda_\Theta \sum_{k=1}^{D} \left\|\Delta_F \Theta^{k\top}\right\|_F^2
\end{aligned}$$

where $A = \begin{bmatrix} \vdots \\ \boldsymbol{\epsilon}^{ij\top} \otimes \mathbf{f}^{ij\top} \\ \vdots \end{bmatrix}$, $\mathbf{b}^k = \begin{bmatrix} \vdots \\ \Delta^i_{kj} \\ \vdots \end{bmatrix}$, $\Delta_F = \begin{bmatrix} \vdots \\ \mathbf{f}^{ij\top} - \mathbf{f}^{ij'\top} \\ \vdots \end{bmatrix}$

To minimize the error:

1. Initialize Θ randomly.

2. Loop:

   (a) For each (i, j), we define $H^{ij} \equiv \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})$ (for fixed Θ^k). We then solve for the best $\boldsymbol{\epsilon}^{ij}$:

$$\begin{aligned} \boldsymbol{\epsilon}^{ij} &= \arg\min_{\{\boldsymbol{\epsilon}^{ij}\}} \left\{ \left\| \mathcal{H}_\theta(\overline{\mathbf{x}}^{ij})\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j} \right\|_2^2 + \lambda_\epsilon \left\| \boldsymbol{\epsilon}^{ij} \right\|_2^2 \right\} && \text{(C.7)} \\ &= \arg\min_{\{\boldsymbol{\epsilon}^{ij}\}} \left\{ \left\| H^{ij}\boldsymbol{\epsilon}^{ij} - \Delta^i_{\cdot j} \right\|_2^2 + \lambda_\epsilon \left\| \boldsymbol{\epsilon}^{ij} \right\|_2^2 \right\} && \text{(C.8)} \\ &= ({H^{ij}}^\top H^{ij} + \lambda_\epsilon I)^+ {H^{ij}}^\top \Delta^i_{\cdot j} && \text{(C.9)} \end{aligned}$$

   (b) For each k, solve for the best Θ^k given the $\boldsymbol{\epsilon}^{ij}$'s:

$$\begin{aligned} \Theta^k &= \arg\min_{\Theta^k} \left\{ \left\| A\text{vec}\left({\Theta^k}^\top\right) - \mathbf{b}^k \right\|_2^2 + \lambda_\Theta \left\| \Delta_F {\Theta^k}^\top \right\|_F^2 \right\} && \text{(C.10)} \\ &= \arg\min_{\Theta^k} \left\{ \left\| A\text{vec}\left({\Theta^k}^\top\right) - \mathbf{b}^k \right\|_2^2 + \lambda_\Theta \left\| (I \otimes \Delta_F)\text{vec}\left({\Theta^k}^\top\right) \right\|_2^2 \right\} && \text{(C.11)} \end{aligned}$$

$$\begin{aligned} \text{vec}\left({\Theta^k}^\top\right) &= (A^\top A + \lambda_\Theta (I \otimes \Delta_F)^\top (I \otimes \Delta_F))^+ A^\top \mathbf{b}^k && \text{(C.12)} \\ &= (A^\top A + \lambda_\Theta (I \otimes ({\Delta_F}^\top \Delta_F))^{-1} A^\top \mathbf{b}^k && \text{(C.13)} \end{aligned}$$

C.4 Derivatives and Hessian for different solving

We want to minimize:

$$\mathrm{err}(\theta, \boldsymbol{\epsilon}^{ij}) = \sum_{k=1}^{D} \left\| A \cdot \mathrm{vec}\left(\Theta^{k\top}\right) - \mathbf{b}^k \right\|_2^2 + \lambda_\epsilon \sum_{i,j \in \mathcal{N}^i} \left\|\boldsymbol{\epsilon}^{ij}\right\|_2^2 + \lambda_\Theta \sum_{k=1}^{D} \left\| (I \otimes \Delta_F)\mathrm{vec}\left(\Theta^{k\top}\right) \right\|_2^2 \quad \text{(C.14)}$$

For that, we can use any gradient descent/LM algorithm using the gradient/Hessian computed as follows:

$$\frac{\partial}{\partial \boldsymbol{\epsilon}^{ij}} \mathrm{err}(\theta, \boldsymbol{\epsilon}^{ij}) = \sum_{k=1}^{D} 2\Theta^k \mathbf{f}^{ij} \left(\mathbf{f}^{ij\top} \Theta^{k\top} \boldsymbol{\epsilon}^{ij} - \Delta^i_{kj} \right) + 2\lambda_\epsilon \boldsymbol{\epsilon}^{ij} \quad \text{(C.15)}$$

$$\frac{\partial}{\partial \mathrm{vec}\left(\Theta^{k\top}\right)} \mathrm{err}(\theta, \boldsymbol{\epsilon}^{ij}) = 2A^\top \left(A \cdot \mathrm{vec}\left(\Theta^{k\top}\right) - \mathbf{b}^k \right) + 2\lambda_\Theta \left(I \otimes \left(\Delta_F{}^\top \Delta_F \right) \right) \mathrm{vec}\left(\Theta^{k\top}\right) \quad \text{(C.16)}$$

And the second order for the Hessian:

$$\frac{\partial^2}{\partial^2 \boldsymbol{\epsilon}^{ij}} \mathrm{err}(\theta, \boldsymbol{\epsilon}^{ij}) = \sum_{k=1}^{D} 2\Theta^k \mathbf{f}^{ij} \mathbf{f}^{ij\top} \Theta^{k\top} + 2\lambda_\epsilon I \quad \text{(C.17)}$$

$$\frac{\partial^2}{\partial^2 \mathrm{vec}\left(\Theta^{k\top}\right)} \mathrm{err}(\theta, \boldsymbol{\epsilon}^{ij}) = 2A^\top A + 2\lambda_\Theta \left(I \otimes \left(\Delta_F{}^\top \Delta_F \right) \right) \quad \text{(C.18)}$$

$$\frac{\partial^2}{\partial \boldsymbol{\epsilon}^{ij} \partial \text{vec}\left(\Theta^{k\top}\right)} = \frac{\partial}{\partial \text{vec}\left(\Theta^{k\top}\right)} 2\Theta^k \mathbf{f}^{ij} \left(\mathbf{f}^{ij\top} \Theta^{k\top} \boldsymbol{\epsilon}^{ij} - \Delta^i_{kj}\right) \tag{C.19}$$

$$= 2\frac{\partial}{\partial \text{vec}\left(\Theta^{k\top}\right)} \Theta^k \mathbf{f}^{ij} \mathbf{f}^{ij\top} \Theta^{k\top} \boldsymbol{\epsilon}^{ij} - 2\frac{\partial}{\partial \text{vec}\left(\Theta^{k\top}\right)} \Delta^i_{kj} B \text{vec}\left(\Theta^{k\top}\right)$$

$$\text{with } B^{ij} = I \otimes \mathbf{f}^{ij\top} \text{ which is } d \times df \tag{C.20}$$

$$= 2\left(\frac{\partial}{\partial \text{vec}\left(\Theta^{k\top}\right)} \left(B^{ij} \text{vec}\left(\Theta^{k\top}\right)\right) \left(B^{ij} \text{vec}\left(\Theta^{k\top}\right)\right)^\top \boldsymbol{\epsilon}^{ij}\right)$$

$$- 2\Delta^i_{kj} B^{ij} \tag{C.21}$$

$$= 2\left(\boldsymbol{\epsilon}^{ij\top} B^{ij} \text{vec}\left(\Theta^{k\top}\right)\right) B^{ij} + 2B^{ij} \text{vec}\left(\Theta^{k\top}\right) \boldsymbol{\epsilon}^{ij\top} B^{ij} - 2\Delta^i_{kj} B^{ij} \tag{C.22}$$

C.5 Manifold Denoising

Assume we have learned or are given the mapping $\mathcal{H}_\Theta$ that describes the tangent space of some manifold or family of manifolds. Suppose we are also given samples $\mathbf{x}^i$ from a manifold that have been corrupted by noise so they no longer lie exactly on the manifold (these can be the same samples from which $\mathcal{H}_\Theta$ was learned). Here we show how to recover a set of points χ^i that are close to the noisy samples $\mathbf{x}^i$ but lie on a manifold consistent with $\mathcal{H}_\Theta$. The key is the simple observation that, as before, if χ^j is a neighbor of χ^i then there exists an unknown $\boldsymbol{\epsilon}^{ij}$ such that $\left\|\mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j)\right\|_2^2$ is small. We can thus find a series of points χ^i that are close to the original points $\mathbf{x}^i$ and satisfy the above. The error is the sum of two terms (λ_{noise} weighs the importance of sticking to the original points):

$$\text{err}_{\text{noise}}(\chi) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j\in\mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j)\right\|_2^2 + \lambda_{\text{noise}} \sum_{i=1}^{n} \left\|\chi^i - \mathbf{x}^i\right\|_2^2 \tag{C.23}$$

$$= \sum_{i,j\in\mathcal{N}^i} \left\|\left(\mathcal{H}_\theta(\overline{\chi}^{ij})\mathcal{H}_\theta(\overline{\chi}^{ij})^+ - I\right)(\chi^i - \chi^j)\right\|_2^2 + \lambda_{\text{noise}} \sum_{i=1}^{n} \left\|\chi^i - \mathbf{x}^i\right\|_2^2 \tag{C.24}$$

We solve the above expression by initially setting $\chi^i = \mathbf{x}^i$ for each i and then performing gradient descent on the χ^i's. We assume that for a small change in χ^i, $\frac{\partial \mathcal{H}_\theta}{\partial \chi^i}$ is negligible with respect to the other quantities. We can therefore define $B^{ij} \equiv \mathcal{H}_\theta(\overline{\chi}^{ij})\mathcal{H}_\theta(\overline{\chi}^{ij})^+ - I$. The derivative w.r.t. χ^i is:

$$\frac{\partial \mathrm{err}_{\text{noise}}(\chi)}{\partial \chi^i} = \frac{\partial}{\partial \chi^i}\left(\sum_{j\in\mathcal{N}^i}\left\|B^{ij}(\chi^i-\chi^j)\right\|_2^2 + \sum_{j|i\in\mathcal{N}^j}\left\|B^{ji}(\chi^j-\chi^i)\right\|_2^2 \right.$$
$$\left. + \lambda_{\text{noise}}\left\|\chi^i-\mathbf{x}^i\right\|_2^2\right) \quad \text{(C.25)}$$

$$= \frac{\partial}{\partial \chi^i}\left(2\sum_{j\in\mathcal{N}^i}\left\|B^{ij}(\chi^i-\chi^j)\right\|_2^2 + \lambda_{\text{noise}}\left\|\chi^i-\mathbf{x}^i\right\|_2^2\right) \quad \text{(C.26)}$$

$$= 4\sum_{j\in\mathcal{N}^i} {B^{ij}}^\top B^{ij}(\chi^i-\chi^j) + 2\lambda_{\text{noise}}\left(\chi^i-\mathbf{x}^i\right) \quad \text{(C.27)}$$

We could solve for the best value of χ^i given χ^j for all $j \neq i$; however, our assumption that $\frac{\partial \mathcal{H}_\theta}{\partial \chi^i}$ is negligible might no longer hold. Instead we simply perform gradient descent by only using the component of the gradient that is orthonormal to the manifold (we cannot correct the noise in the colinear direction). We find a (possibly local) minimum of $\mathrm{err}_{\text{noise}}$ by iterating until convergence.

C.6 Snakes on a Local Plane

We are given two points on a manifold, $\mathbf{x}$ and $\mathbf{x}'$, and we want to find the geodesic distance between them. We will find a minimal path between these two points by employing an active contour model (snake). The snake will be composed of points $\chi^1, \ldots, \chi^m$, where $\chi^1 = \mathbf{x}$ and $\chi^m = \mathbf{x}'$. It will have to obey the following constraints:

1. The overall length of the snake should be minimal. The length is given by $\sum_{i=2}^{m} \|\chi^i - \chi^{i-1}\|_2$. However, this sum is hard to work with, so for computational reasons, we will use the following approximation instead:

$$\text{err}_{\text{length}}(\chi) = \sum_{i=2}^{m} \left\|\chi^i - \chi^{i-1}\right\|_2^2 \tag{C.28}$$

2. As in the previous section, the points should be consistent with $\mathcal{H}_\Theta$. Here, each point i that is not an extremity of the snake has a neighborhood: $\mathcal{N}^i = \{i-1, i+1\}$. This constraint can be expressed as the minimization of:

$$\text{err}_{\mathcal{M}}(\chi) = \min_{\{\boldsymbol{\epsilon}^{ij}\}} \sum_{i,j \in \mathcal{N}^i} \left\|\mathcal{H}_\theta(\overline{\chi}^{ij})\boldsymbol{\epsilon}^{ij} - (\chi^i - \chi^j)\right\|_2^2 \tag{C.29}$$

$$= \sum_{i,j \in \mathcal{N}^i} \left\|B^{ij}(\chi^i - \chi^j)\right\|_2^2 \tag{C.30}$$

The derivatives of these energies should be 0 for the fixed extremity points $i = 1$ and $i = m$. For the points in between:

$$\frac{\partial \text{err}_{\text{length}}}{\partial \chi^i}(\chi) = \frac{\partial}{\partial \chi^i}\left(\left\|\chi^i - \chi^{i-1}\right\|_2^2 + \left\|\chi^{i+1} - \chi^i\right\|_2^2\right) \tag{C.31}$$

$$= \sum_{j=i\pm1} \frac{\partial}{\partial \chi^i}\left\|\chi^i - \chi^j\right\|_2^2 = 2\sum_{j=i\pm1}(\chi^i - \chi^j) \tag{C.32}$$

$$\frac{\partial \text{err}_{\mathcal{M}}}{\partial \chi^i}(\chi) = 4\sum_{j \in \mathcal{N}^i} {B^{ij}}^\top B^{ij}(\chi^i - \chi^j) \tag{C.33}$$

We use the following algorithm to minimize the length of the snake while keeping it on the manifold:

- The snake is initialized by finding a path between x and x' using Dijkstra's algorithm on points from the original dataset. These points form an initial snake. Then, depending on the accuracy wanted for the result, extra points are added between the existing points by linear interpolation.

- This initial snake is first optimized to be as much as possible on the manifold. This is done by minimizing $\text{err}_{\mathcal{M}}$ using gradient descent but by forcing the direction of minimization to be orthogonal to the gradient field on the manifold.

- The length of the snake is then optimized by minimizing $\text{err}_{\text{length}}$ using gradient descent too, this time by forcing the direction of minimization to be colinear to the gradient field on the manifold. To prevent drifting, the snake is sometimes "brought back" onto the manifold using the previous step.

Appendix D

Non-Linear Embedding Formulas

D.1 Best N-view Reconstruction

We want to solve for:

$$\begin{aligned} \min_{S,R_t} \quad & \sum_{t=1}^{f} \|W_t - R_t S\|_F^2 && \text{(D.1)} \\ \text{subject to} \quad & R_t \text{ is the top two rows of a rotation matrix} && \text{(D.2)} \\ & S \text{ is centered} && \text{(D.3)} \\ & W_t \text{ is centered} && \text{(D.4)} \\ & R_1 = \Pi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} && \text{(D.5)} \end{aligned}$$

The equation can be re-written as:

$$\begin{aligned} \min_{R_t,S} \quad & f(S,R_t) = \sum_{t=1}^{f} \|W_t \mathcal{C}_{n-1} - R_t S \mathcal{C}_{n-1}\|_F^2 && \text{(D.6)} \\ \text{subject to} \quad & R_t \text{ is the top two rows of a rotation matrix} && \text{(D.7)} \\ & R_1 = \Pi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} && \text{(D.8)} \end{aligned}$$

by using the same $\mathcal{C}_{n-1}$ as in Section B.1, and shortening W_t's and S by one element. We have:

$$\frac{\partial}{\partial S} f(S, R_t) = \sum_{t=1}^{f} -2R_t^\top (W_t\mathcal{C}_{n-1} - R_t S\mathcal{C}_{n-1})\mathcal{C}_{n-1}^\top = -2\sum_{t=1}^{f} R_t^\top (W_t - R_t S)\mathcal{C}_{n-1}\mathcal{C}_{n-1}^\top \tag{D.9}$$

As $\mathcal{C}_{n-1}\mathcal{C}_{n-1}^\top$ is invertible and as the expression above is null at the optimum, we have there:

$$S_{\text{opt}} = \left(\sum_{t=1}^{f} R_t^\top R_t\right)^{-1} \sum_{t=1}^{f} R_t^\top W_t \tag{D.10}$$

D.2 Simplifying the problem for 2 views

If $f = 2$ in $f(S, R_t)$, the first rotation is Π and let the second rotation be R. We now have the two terms in the sum at the optimum in R:

$$\begin{aligned}
\|W_1 - \Pi S_{\text{opt}}\|_F^2 &= \left\| W_1 - \Pi \left(\Pi^\top \Pi + R^\top R\right)^{-1} \left(\Pi^\top W_1 + R^\top W_2\right)\right\|_F^2 && \text{(D.11)}\\
&= \left\| \left(\mathrm{I} - \Pi A \Pi^\top\right) W_1 - \Pi A R^\top W_2 \right\|_F^2 && \text{(D.12)}\\
\|W_2 - R S_{\text{opt}}\|_F^2 &= \left\| W_2 - R \left(\Pi^\top \Pi + R^\top R\right)^{-1} \left(\Pi^\top W_1 + R^\top W_2\right)\right\|_F^2 && \text{(D.13)}\\
&= \left\| -RA\Pi^\top W_1 + \left(\mathrm{I} - RAR^\top\right) W_2 \right\|_F^2 && \text{(D.14)}
\end{aligned}$$

with $A = \left(\Pi^\top \Pi + R^\top R\right)^{-1}$. A rotation can be found to transform the first equation in the norm to the second one (this can be done by expressing R in its quaternion form and solving for such a matrix). I could only find such a solution throught the symbolic toolbox of Matlab. If the quaternion of R is $\left[a, b, c, d\right]$, then, this rotation matrix is simply:

$$\frac{1}{a^2 + d^2} \begin{bmatrix} d^2 - a^2 & 2ad \\ -2ad & d^2 - a^2 \end{bmatrix} \tag{D.15}$$

Therefore, the problem now becomes:

$$\min_R \quad f(R) = 2 \left\| \left(\mathrm{I} - \Pi A \Pi^\top\right) W_1 - \Pi A R^\top W_2 \right\|_F^2 \tag{D.16}$$

$$\text{subject to} \quad R \text{ is the top two rows of a rotation matrix} \tag{D.17}$$

Let us consider the rotation $\overline{R}^* = R^{*\top}$. Then it can easily be shown that:

$$\left(\mathbf{I} - \Pi A \Pi^\top\right) = -\frac{1}{2}\left(\frac{1}{\overline{R}_{\cdot 3}^\top \overline{R}_{\cdot 3}} \overline{R}_{\cdot 3} \overline{R}_{\cdot 3}^\top - \mathbf{I}\right) \tag{D.18}$$

$$\Pi A R^\top = -\frac{1}{2}\left(-\left(\frac{1}{\overline{R}_{\cdot 3}^\top \overline{R}_{\cdot 3}} \overline{R}_{\cdot 3} \overline{R}_{\cdot 3}^\top - \mathbf{I}\right) \overline{R}_{\cdot,1:2}\right) \tag{D.19}$$

Therefore, solving for this problem is equivalent to solving for the infimum in Section B.1, the minimum being twice smaller.

D.3 Gradient of the error

Let us recall the equation of the error to optimize in MSFM (*cf*.Equation (4.4)):

$$\min_{R_t, \mathbf{t}_t, S_t} \sum_{t=0}^{f} \left\| \left(R_t S_t + \mathbf{t}_t \mathbf{1}_n{}^\top\right) - W_t \right\|_F^2 + \lambda_S \sum_{t=2}^{f} \|S_t - S_{t-1}\|_F^2 + \lambda_R \sum_{t=2}^{f} \|R_t - R_{t-1}\|_F^2 + \lambda_{\mathbf{t}} \sum_{t=2}^{f} \|\mathbf{t}_t - \mathbf{t}_{t-1}\|_F^2 \tag{D.20}$$

D.3.1 Gradient with respect to $\mathbf{t}_i$

The derivative of Equation (4.4) with respect to $\mathbf{t}_i$ is:

$$\frac{\partial}{\partial \mathbf{t}_i} \mathrm{err}_{\mathrm{MSFM}} = \frac{\partial}{\partial \mathbf{t}_i}\left(\sum_{t=1}^{f} \left\| \left(R_t S_t + \mathbf{t}_t \mathbf{1}_n{}^\top\right) - W_t \right\|_F^2 + \lambda_{\mathbf{t}} \sum_{t=2}^{f} \|\mathbf{t}_t - \mathbf{t}_{t-1}\|_2^2\right) \tag{D.21}$$

$$= 2(R_i S_i + \mathbf{t}_i \mathbf{1}_n{}^\top - W_i)\mathbf{1}_n{}^\top + 2\lambda_{\mathbf{t}}\left(\mathbf{t}_i - \mathbf{t}_{i-1} + \mathbf{t}_i - \mathbf{t}_{i+1}\right) \tag{D.22}$$

(D.23)

D.3.2 Gradient with respect to R_i

The derivative of Equation (4.4) with respect to R_i is:

$$\frac{\partial}{\partial R_i}\mathrm{err}_{\mathrm{MSFM}} = \frac{\partial}{\partial R_i}\sum_{t=0}^{f}\left\|\left(R_t S_t + \mathbf{t}_t \mathbf{1}_n{}^\top\right) - W_t\right\|_F^2 + \lambda_R \sum_{t=2}^{f}\left\|R_t - R_{t-1}\right\|_F^2 \quad \text{(D.24)}$$

$$= 2\left(R_i S_i + \mathbf{t}_i \mathbf{1}_n{}^\top - W_i\right) S_i^\top + 2\lambda_R\left(2R_i - R_{i-1} - R_{i+1}\right) \quad \text{(D.25)}$$

$$\text{(D.26)}$$

We then use the composition of derivation rule to get the derivation with respect to the quaternions.

D.3.3 Gradient with respect to S_i

The derivative of Equation (4.4) with respect to S_i is:

$$\frac{\partial}{\partial S_i}\mathrm{err}_{\mathrm{MSFM}} = \frac{\partial}{\partial S_i}\sum_{t=0}^{f}\left\|\left(R_t S_t + \mathbf{t}_t \mathbf{1}_n{}^\top\right) - W_t\right\|_F^2 + \lambda_R \sum_{t=2}^{f}\left\|S_t - S_{t-1}\right\|_F^2 \quad \text{(D.27)}$$

$$= 2R_i^\top\left(R_i S_i + \mathbf{t}_i \mathbf{1}_n{}^\top - W_i\right) + 2\lambda_S\left(S_i - S_{i-1} + S_i - S_{i+1}\right) \quad \text{(D.28)}$$

$$\text{(D.29)}$$

References

Agarwal, S., Snavely, N., and Seitz, S., 2008: Fast algorithms for l_∞ problems in multi-view geometry. In *IEEE Computer Vision and Pattern Recognition or CVPR*.

Agarwal, S., Wills, J., Cayton, L., Lanckriet, G., Kriegman, D., and Belongie, S., 2007: Generalized non-metric multidimensional scaling. In *AISTATS*.

Bengio, Y., and Monperrus, M., 2005: Non-local manifold tangent learning. In *NIPS*.

Bengio, Y., Paiement, J., Vincent, P., Delalleau, O., Le Roux, N., and Ouimet, M., 2004: Out-of-sample extensions for LLE, isomap, MDS, eigenmaps, and spectral clustering. In *NIPS*.

Blake, A., and Isard, M., 1998: *Active Contours*. Springer, Berlin Heidelberg New York.

Brand, M., 2003: Charting a manifold. In *NIPS*.

Brand, M., 2005: A direct method for 3d factorization of nonrigid motion observed in 2d. In *CVPR*, 122– 128.

Brand, W., 2001: Morphable 3d models from video. In *CVPR*, II–456– II–463.

Bregler, C., Hertzmann, A., and Biermann, H., 2000: Recovering non-rigid 3d shape from image streams. In *CVPR*, 690–696.

Carceroni, R. L., Padua, F. L. C., Santos, G. A. M. R., and Kutulakos, K. N., 2004: Linear sequence-to-sequence alignment. *CVPR*, **01**, 746–753. ISSN 1063-6919. doi: http://doi.ieeecomputersociety.org/10.1109/CVPR.2004.150.

Chandraker, M., and Kriegman, D., 2008: Globally optimal bilinear programming for computer vision applications. In *IEEE Computer Vision and Pattern Recognition or CVPR*.

Chen, P., and Suter, D., 2004: Recovering the missing components in a large noisy low-rank matrix: Application to SFM. *IEEE Trans. Pattern Analysis and Machine Intelligence*, **26**(8), 1051–1063.

Cheung, K., Baker, S., and Kanade, T., 2005: Shape-from-silhouette across time part i: Theory and algorithms. *International Journal of Computer Vision*, **62**(3), 221–247.

Cootes, T., Edwards, G., and Taylor, C., 2001: Active appearance models. In *PAMI*, 6, 681–685.

Costeira, J. P., and Kanade, T., 1998: A multibody factorization method for independently moving objects. In *IJCV*, volume 29.

Del Bue, A., Smeraldi, F., and Agapito, L., 2007: Non-rigid structure from motion using ranklet-based tracking and non-linear optimization. In *IVC*, volume 25, 297–310.

DeMers, D., and Cottrell, G., 1993: Non-linear dimensionality reduction. In *NIPS*.

Dollár, P., Rabaud, V., and Belongie, S., 2006: Learning to traverse image manifolds. In *NIPS*, volume 19.

Dollár, P., Rabaud, V., and Belongie, S., 2007: Non-isometric manifold learning: Analysis and an algorithm. In *ICML*.

Donoho, D., and Grimes, C., 2003: Hessian eigenmaps: locally linear embedding techniques for high dimensional data. *Proc. of National Academy of Sciences*.

Elgammal, A., and Lee, C., 2004: Separating style and content on a nonlinear manifold. In *CVPR*.

Faugeras, O., and Luong, Q., 2004: *The Geometry of Multiple Images: The Laws That Govern the Formation of Multiple Images of a Scene and Some of Their Applications*. MIT Press.

Fischler, M. A., and Bolles, R. C., 1981: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, **24**(6), 381–395. ISSN 0001-0782. doi:10.1145/358669.358692.

Hastie, T., Tibshirani, R., and Friedman, J., 2001: *The Elements of Statistical Learning*. Springer.

Henrion, D., and Lasserre, J.-B., 2003: Gloptipoly: Global optimization over polynomials with matlab and sedumi. *ACM Trans. Math. Softw.*, **29**(2), 165–194. ISSN 0098-3500. doi:http://doi.acm.org/10.1145/779359.779363.

Horn, B., 1986: *Robot Vision*. MIT Press.

Horn, B., 1987: Closed form solutions of absolute orientation using orthonormal matrices. In *Journal of the Optical Society of America*, volume 5, 1127–1135.

Jacobs, D., 1997: Linear fitting with missing data: Applications to structure from motion and to characterizing intensity images. In *IEEE Computer Vision and Pattern Recognition or CVPR*, 206–212.

Kahl, F., 2005: Multiple view geometry and the l_∞-norm. In *International Conference on Computer Vision*, II: 1002–1009.

Kass, M., Witkin, A., and Terzopoulos, D., 1988: Snakes: Active contour models. *IJCV*.

Keysers, D., Macherey, W., Dahmen, J., and Ney, H., 2001: Learning of variability for invariant statistical pattern recognition. *ECML*.

Kim, T., and Hong, K.-S., 2005: Estimating approximate average shape and motion of deforming objects with a monocular view. In *IJPRAI*, volume 19, 585–601.

Kutulakos, K., and Seitz, S., 2000: A theory of shape by space carving. *International Journal of Computer Vision*, **38**(3), 199–218.

Laptev, I., Belongie, S. J., Perez, P., and Wills, J., 2005: Periodic motion detection and segmentation via approximate sequence alignment. In *ICCV*, volume 1, 816–823.

Laptev, I., and Perez, P., 2007: Retrieving actions in movies. In *ICCV*, 1–8.

Lepetit, V., Moreno-Noguer, F., and Fua, P., 2008: Epnp: An accurate o(n) solution to the pnp problem. In *International Journal of Computer Vision*.

Li, H., 2007: A practical algorithm for l triangulation with outliers. 1–8.

Llado, X., Del Bue, A., and Agapito, L., 2005: Non-rigid 3d factorization for projective reconstruction. In *BMVC*.

Lourakis, M., and Argyros, A., 2004: The design and implementation of a generic sparse bundle adjustment software package based on the levenberg-marquardt algorithm. Technical Report 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece. Available from `http://www.ics.forth.gr/~lourakis/sba`.

Miller, M., and Younes, L., 2001: Group actions, homeomorphisms, and matching: A general framework. *IJCV*.

Park, J. H., Zhang, Z., Zha, H., and Kasturi, R., 2004: Local smoothing for manifold learning. In *CVPR*.

Rao, R., and Ruderman, D., 1999: Learning Lie groups for invariant visual perception. In *NIPS*.

Roweis, S. T., and Saul, L. K., 2000: Nonlinear dimensionality reduction by locally linear embedding. *Science*, **290**.

Saul, L. K., and Roweis, S. T., 2003: Think globally, fit locally: unsupervised learning of low dimensional manifolds. *JMLR*.

Schölkopf, B., Smola, A., and Müller, K., 1998: Nonlinear component analysis as a kernel eigenvalue problem. *Neural Information Processing Systems*.

Schweikert, D. G., and Kernighan, B. W., 1972: A proper model for the partitioning of electrical circuits. In *DAC '72: Proceedings of the 9th workshop on Design automation*, 57–62. ACM, New York, NY, USA. doi:http://doi.acm.org/10.1145/800153.804930.

Shi, J., and Malik, J., 2000: Normalized cuts and image segmentation. In *PAMI*, 8, 888–905.

Simard, P., LeCun, Y., Denker, J. S., and Victorri, B., 1998: Transformation invariance in pattern recognition-tangent distance and tangent propagation. In *Neural Networks: Tricks of the Trade*.

Soatto, S., and Perona, P., 1994: Recursive estimation of camera motion from uncalibrated image sequences. In *ICIP*, III: 58–62.

Tenenbaum, J. B., de Silva, V., and Langford, J. C., 2000: A global geometric framework for nonlinear dim. reduct. *Science*, **290**.

Tenenbaum, J. B., and Freeman, W. T., 2000: Separating style and content with bilinear models. *Neural Computation*, **12**(6).

Tomasi, C., and Kanade, T., 1992: Shape and motion from image streams under orthography: a factorization method. In *IJCV*, volume 9.

Torresani, L., Hertzmann, A., and Bregler, C., 2003: Learning non-rigid 3d shape from 2d motion. In *NIPS*.

Torresani, L., Hertzmann, A., and Bregler, C., 2008: Non-rigid structure-from-motion: Estimating shape and motion with hierarchical priors. In *PAMI*.

Torresani, L., Yang, D., Alexander, E., and Bregler, C., 2001: Tracking and modeling non-rigid objects with rank constraints. In *CVPR*, I–493– I–500.

Triggs, B., McLauchlan, P. F., Hartley, R. I., and Fitzgibbon, A. W., 2000: Bundle adjustment - a modern synthesis. In *ICCV*, 298–372. Springer-Verlag, London, UK. ISBN 3-540-67973-1.

Vidal, R., and Abretske, D., 2006: Nonrigid shape and motion from multiple perspective views. In *ECCV*, II: 205–218.

Weinberger, K. Q., and Saul, L. K., 2006: Unsupervised learning of image manifolds by semidefinite programming. *International Journal of Computer Vision*.

Xiao, J., Chai, J., and Kanade, T., 2006a: A closed-form solution to non-rigid shape and motion recovery. In *IJCV*, volume 67.

Xiao, J., and Kanade, T., 2004: Non-rigid shape and motion recovery: degenerate deformations. In *CVPR*, I–668– I–675 Vol.1.

Xiao, J., and Kanade, T., 2005: Uncalibrated perspective reconstruction of deformable structures. In *ICCV*, 1075– 1082 Vol. 2.

Xiao, L., Sun, J., and Boyd, S., 2006b: A duality view of spectral methods for dimensionality reduction. In *ICML*.

Yan, J., and Pollefeys, M., 2005: A factorization-based approach to articulated motion recovery. In *CVPR*, 815– 821.