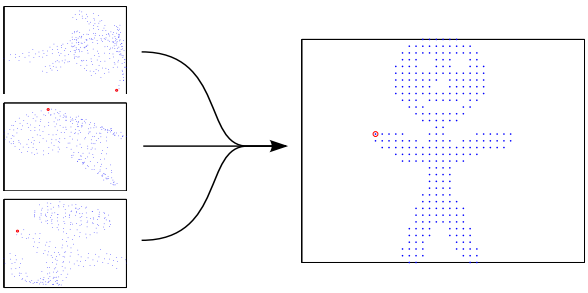
UNIVERSIDADE TÉCNICA DE LISBOA

INSTITUTO SUPERIOR TÉCNICO



Reconstruction Of Isometrically Embedded Flat Surfaces From Scaled Orthographic Image Data

Ricardo Jorge dos Santos Ferreira

(Mestre)

Dissertação para a Obtenção do Grau de Doutor em

Engenharia Electrotécnica e de Computadores

Orientador: Doutor João Paulo Salgado Arriscado Costeira

Co-Orientador: Doutor João Manuel de Freitas Xavier

Júri

Presidente: Presidente do Conselho Científico do IST

Vogais: Doutor Carlo Tomasi

Doutora Maria de Fátima Silva Leite

Doutor Diogo Luís de Castro Vasconcelos de Aguiar Gomes

Doutor Jorge dos Santos Salvador Marques

Doutor João Paulo Salgado Arriscado Costeira

Doutor João Manuel de Freitas Xavier

Maio 2010

Resumo

Nesta tese propõe-se um método para reconstrução de uma classe de superfícies não rígidas a partir de características pontuais em imagens. A classe de superfícies às quais o método se aplica consiste em superfícies de curvatura nula, mergulhadas suavemente no espaço 3D Euclidiano em diversas configurações. Um exemplo tipicamente usado consiste numa folha de papel curvado suavemente em diversas configurações. Por reconstrução entende-se recuperar: 1) a posição das características visíveis quando a superfície é desenrolada; 2) a pose tridimensional da superfície em cada imagem observada. As câmaras consideradas são do tipo ortográfico com escala e não se assume a prévia calibração destas. Assume-se que os pontos característicos observados em cada imagem foram previamente emparelhados entre imagens, mas possibilita-se a existência de oclusões.

Dada a complexidade em representar funções isometricas, não é possível descrever uma função de custo simples e assumir que existe um método de optimização que a resolve. Assim o problema é separado em vários sub-problemas, onde cada um refina a solução dada pelo anterior. Os primeiros destes sub-problemas encontram uma solução discreta, ou seja onde se consideram apenas os pontos característicos dados. O último passo porém desenvolve uma ponte que permite a passagem para um modelo contínuo.

A solução depende fortemente de uma classe de matrizes não usadas correntemente, aqui designadas como “sub-Stiefel”. Estas matrizes são de grande importância quando se consideram câmaras ortográficas e ortográficas com escala, mas não se encontra qualquer literatura sobre este conjunto. Aqui, estas matrizes são caracterizadas e descritas em profundidade.

Palavras Chave: Visão Por Computador, Reconstrução Tri-Dimensional, Reconstrução Não Rígida, Estrutura a Partir de Movimento, Mergulhos Isométricos, Optimização Não Linear

Abstract

In this thesis a method to reconstruct a class of non-rigid surfaces from image point features is presented. The class of surfaces to which it applies consists of flat surfaces isometrically and smoothly embedded in Euclidean three-space of which the model example is a smoothly bent sheet of paper observed in different configurations. Here it is proposed to recover: 1) the feature locations of the flattened surface as well; 2) the three-dimensional pose of the surface in each image. The cameras are considered to be scaled orthographic and they are not assumed to be previously calibrated. It is assumed that the features have been previously matched between images but occlusions are allowed.

Due to the complexity of representing isometric functions it is not possible to describe a simple cost function and assume that there's an optimization algorithm that solves it. Instead, the problem is split into subproblems, where each step refines the previously obtained solution. The first subproblems deal with finding a discrete solution for the problem, i.e. one where only the feature points are considered. The last step provides a bridge that allows for the whole continuous embedded surface to be considered.

The solution depends heavily on certain non-mainstream matrices, here denoted as "sub-Stiefel". These matrices are of great importance when considering orthographic and scaled orthographic cameras, but no literature describing them has been found so far. Here, these matrices are characterized and described in depth.

Keywords: Computer Vision, 3D Reconstruction, Non Rigid Reconstruction, Structure From Motion, Isometric Embeddings, Non Linear Optimization

Contents

List of Figures

List of Tables

List of Symbols

$\mathbb{R}^+$	Set of positive real numbers	8
$\mathbf{C}_i^k$	The Jacobian matrix of the camera projection function C^k at a point $\mathbf{r}_i^k$	9
$\mathsf{M}^{m\times n}$	The set of $m \times n$ matrices	9
$\mathbb{T}_{\mathbf{q}_i}\mathbb{R}^2$	The tangent space of $\mathbb{R}^2$ at the point $\mathbf{q}_i$	9
$\mathbb{T}_{\mathbf{r}_i^k}\mathcal{S}^k$	The tangent space of $\mathcal{S}^k$ at the point $\mathbf{r}_i^k$	9
$\mathbb{T}_{\mathbf{p}_i^k}\mathbb{R}^2$	The tangent space of $\mathbb{R}^2$ at the point $\mathbf{p}_i^k$	9
$\mathbf{w}_i$	A tangent vector at $\mathbf{q}_i$	9
$\mathbf{u}_i^k$	A tangent vector at $\mathbf{r}_i^k$	9
$\mathbf{v}_i^k$	A tangent vector at $\mathbf{p}_i^k$	9
s^k	The kth camera scale factor	9
$\mathbf{P}^k$	The kth camera orthographic projection matrix (Stiefel)	9
$\mathbf{V}_i$	Matrix containing the observed neighbor vectors at point i in all images	14
$\mathbf{U}_i$	Matrix containing the reconstructed 3D vectors at point i	14
$\mathbb{GL}(n)$	The set of $n \times n$ invertible matrices.	15
$\mathbf{J}_i^k$	Jacobian matrices of the isometry functions $\mathcal{I}^k$ at each point $\mathbf{q}_i$	22
$\mathbb{SS}$	The sub-Stiefel set	22
$\mathbf{O}_i^k$	Sub-Stiefel matrix used to describe de Jacobian matrix at point $\mathbf{q}_i$ of $C^k \circ \mathcal{I}^k$	23
$\mathbf{M}_i$	Matrix containing the Jacobian matrices of $C^k \circ \mathcal{I}^k$ in all images	23
$\mathbf{W}_i$	Matrix containing the reconstructed 3D vectors at point i	23
$\mathbf{1}_n$	The n dimensional vector filled with ones	30
$\mathbf{I}_n$	The $n \times n$ identity matrix	30
$\mathbf{o}_i^k$	The completion of a sub-Stiefel matrix to a Stiefel matrix	38
x_i^k	First coordinate of $\mathbf{r}_i^k$	39
y_i^k	Second coordinate of $\mathbf{r}_i^k$	39
z_i^k	Third coordinate of $\mathbf{r}_i^k$	39
a_i^k	Unknown sign ambiguity of $\mathbf{o}_i^k$	39
r	An isometric embedding function	50
c	A function defining a $3D$ path needed for defining an isometry	50
d	A function defining the $3D$ direction known as the directrix of the surface	50
l_t	Function describing the line segments known as the rulings of a surface	50
α	Function defining the limits of the rulings	50
β	Function defining the limits of the rulings	50

Chapter 1

Introduction

Contents

Inferring structure from image data has been one of the objectives of Computer Vision from the beginning. Classical algorithms assume a rigid scene observed by different cameras and attempt to obtain a 3D computer description from different features (e.g. color, corners, shade, etc). While these have been studied to exhaustion, only recently has reconstruction of scenes which differ between image frames become main-stream, probably motivated by the exponential increase in available processing capabilities which allows for ever more complex algorithms to be executed in reasonable time.

Once the rigidity requirement is lifted, several new classes of problems arise. One can consider scenes with multiple rigid bodies (e.g. a robot arm consisting of several articulations), scenes which are allowed to bend and/or stretch (e.g. a tree waving at the wind), scenes with strange dynamics (e.g. fire), and many other variants.

The proposed thesis deals with the reconstruction of surfaces which are allowed to deform in such a way that intrinsic distances are preserved (i.e. no stretching or shearing is allowed). A sheet of paper waving is the prototype example, but several types of cloth which are rigid enough not to shear or stretch may be considered as well.

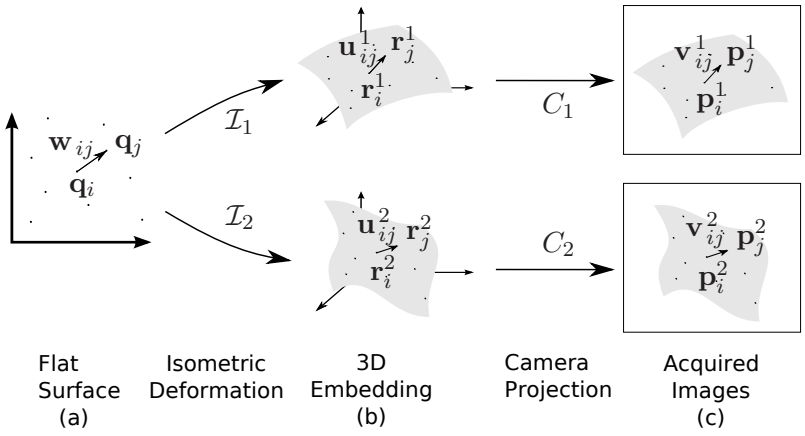


Figure 1.1: Acquisition model of the isometrically embedded surface observed by different cameras in different points in space and time.

1.1 Problem Statement

The earliest design decision introduced is that the object to be reconstructed is represented by a finite set of feature points, usually inferred from texture. These feature points are assumed to be dense enough with respect to the amount of deformation. The set of feature points on the unfolded surface is here denoted by $\mathcal{Q} = \{\mathbf{q}_i\}_1^N \subset \mathcal{U} \subset \mathbb{R}^2$, where each $\mathbf{q}_i$ denotes a single 2D surface point. The notation used means that the elements of the set are indexed from 1 to N, the number of elements in the set. The subset $\mathcal{U}$ is used to limit the extents of the continuous surface and for practical purposes it can be assumed to be bounded since all physical surfaces must be finite. If no other information other than the point cloud is provided, it is considered to be the convex hull of the point cloud.

The object features are then embedded in specific 3D poses by a set of embedding functions $\left\{\mathcal{I}^k : \mathcal{U} \subset \mathbb{R}^2 \to \mathbb{R}^3\right\}_1^K$. Notice the same notation being used once again to state that the elements of this set are indexed from 1 to K. The surface spanned by the image of each of these functions is called an **embedding pose** and will be denoted by $\mathcal{S}^k = \mathcal{I}^k(\mathcal{U}) \subset \mathbb{R}^3$. Of the whole surface, only a finite number of feature points are considered, here collected in the set $\mathcal{R}^k = \mathcal{I}^k(\mathcal{Q}) \subset \mathcal{S}^k$. The elements of this set are numbered in the same order as the points $\mathbf{q}_i$ so $\mathcal{R}^k = \left\{\mathbf{r}_i^k\right\}_{i=1}^N$ such that $\mathbf{r}_i^k = \mathcal{I}^k(\mathbf{q}_i)$. As previously stated, the embedding functions $\mathcal{I}^k$ must obey several constraints so as not to shear or stretch the original 2D surface. For the sake of clarity, the full characterization of these functions is delayed.

Finally each of the embedded point sets $\mathcal{R}^k$ is observed by a different camera which projects the 3D feature points to a 2D image using camera projection functions $\left\{C^k : \mathbb{R}^3 \to \mathbb{R}^2\right\}_1^K$. Each of these projections generates a 2D feature point cloud $\mathcal{P}^k = C^k(\mathcal{R}^k) \subset \mathbb{R}^2$ which is again numbered

in accordance to the previously chosen order $\mathcal{P}^k = \left\{\mathbf{p}_i^k\right\}_{i=1}^N$ such that $\mathbf{p}_i^k = C^k(\mathbf{r}_i^k)$. Figure 1.1 summarizes the acquisition model, which is algebraically stated as

$$\mathbf{p}_i^k = C^k\left(\mathcal{I}^k\left(\mathbf{q}_i\right)\right) \quad \forall i,k\,. \tag{1.1}$$

This equation is generic enough to be the starting point of several computer vision problems and is more or less difficult to solve, depending on the constraints imposed and the available observed data. In almost every case, when applied to all available data the equations define an over-constrained system meaning that no solution satisfies them exactly in real-world applications due to the presence of noise and sensor limitations. As a work-around these equations are often "solved" by a least-squares optimization problem formulation. When done correctly, this finds the best solution (in a certain very specific sense) that almost-fits. Often it also happens that these equations should admit more than one solution when no noise is present, meaning that the problem is also ill-posed. In these cases the geometry of the problem allows a continuum of solutions to be identified, but the presence of noise might mislead by allowing for a single solution to be identified if the formulation does not take this design constraint into account. This single solution is meaningless with respect to others and is often very ill conditioned.

A particularly successful historical example can be formulated when orthographic cameras are considered and all embedding functions $\mathcal{I}^k$ are constrained to be the same ($\mathcal{I}^k = \mathcal{I}$). This implies that all sets $\mathcal{R}^k$ are equal and that, when zero-centered data is considered, the functions C^k are linear functions represented as Stiefel matrices $\mathbf{C}^k \in \mathsf{O}(2,3)$. The Stiefel set $\mathsf{O}(2,3)$ is simply the set of 3D rotations where the last line has been erased. The equation thus simplifies to

$$\mathbf{p}_i^k = \mathbf{C}^k \underbrace{\mathcal{I}\left(\mathbf{q}_i\right)}_{\mathbf{r}_i} \quad \forall i,k. \tag{1.2}$$

The classic Tomasi-Kanade algorithm [41] is able to recover a meaningful solution consisting of matrices $\mathbf{C}^k$ and points $\mathbf{r}_i \in \mathbb{R}^3$ from the observations $\mathbf{p}_i^k \in \mathcal{P}^k$. As discussed above, given enough images the problem is over-constrained, and if ($\hat{\mathbf{C}}^k$, $\hat{\mathbf{r}}_i$) is a solution to the problem then any rotation matrix $\mathbf{R} \in \mathsf{O}(3)$ generates a new, equally valid solution ($\hat{\mathbf{C}}^k\mathbf{R}$, $\mathbf{R}^T\hat{\mathbf{r}}_i$).

Unlike the Tomasi-Kanade example, in this thesis it is not considered that the embedding functions $\mathcal{I}^k$ are the same for all k. Instead, they are assumed to belong to a class of functions known as "isometries", adding a new difficulty that must first be identified and represented and later overcome. Representing the isometry functions correctly is not a trivial task and this work will focus on

two distinct approaches, one simpler (but coarser), the other slightly more complex which is able to completely represent these isometry functions. For the sake of clarity, the simpler description will be given first, while delaying the more complex description to chapter 5.

Looking at equation (1.1), three problems are formulated to be addressed in this thesis:

1. Recover the surface points $\mathcal{Q}$ solely from the multiple observations $\mathcal{P}^k$. In figure 1.1 this means from several observations (c), obtain (a). This problem is here called the **surface unfolding** problem and will be explored in chapter 3.

2. Estimate the embedded points $\mathcal{R}^k$ if the observations $\mathcal{P}^k$ and the underlying surface points $\mathcal{Q}$ are known. In figure 1.1 this means estimate (b) given (a) and (c). This problem is here called the **pose estimation** problem and will be addressed in chapter 4.

3. Represent and estimate the embedding functions $\mathcal{I}^k$ and use this information to improve the already obtained solutions. This problem is here called the **isometry estimation** problem and is discussed in chapter 5.

Prior to exploring the solution to the stated problems, some initial considerations are needed which are provided in chapter 2.

1.2 State of the Art

The first known attempts at fitting developable surfaces to point clouds are described in the papers [31, 18, 4]. The importance of these papers to this thesis was somewhat shadowed by the book [32] which shares a common author with the papers. Chapter 6 and, to a lesser extent, chapter 5 of this book constitutes the primary source of information on developable surfaces used in this thesis. Other book resources that cover the topic include sections 3-5 and 5-8 of [7] and chapters 3 and 5 of the third book in the series [38].

In [10] the importance of developable surfaces in ship-hull design is stated, since it allows for easy manufacturing without stretching or tearing and without the use of heat treatment. The paper itself focuses on approximating developable surfaces using B-splines, coining the term “quasi-developable”, for use in computer aided design. The result is not exactly developable, but the authors claim is a good enough approximation for engineering purposes due to metal plasticity. The intent of representing developable surfaces within a computer is also the main motivation of [40], but now for representing everyday objects in computer graphics. Instead of providing an approximation to developable surfaces,

these authors consider a subclass of developable surfaces to approximate a general one. They do so by considering a piecewise approximation by generalized cones which are a particular kind of developable surfaces. In [28] a technique for reconstructing a 3D developable surface from a 3D point cloud is described. They describe developable surfaces as a 1 parameter family of tangent planes and make these tangent planes agree with the point cloud data. Using tangent plane representation of developable surfaces is known as the dual representation (see [32]) and has the advantage that the resulting surface is guaranteed to be developable. A different technique described in [20] uses triangle meshes to approximate developable surfaces and for computing the development (unfolding) of these surfaces.

Although not focused specifically on developable surfaces, in [23] techniques for smooth interpolation of ruled surfaces are presented. Since ruled surfaces are a superset of developable surfaces, the content provides insight which can be used for the later class.

In [2] the state of the art in developable surfaces is cited to be [11] which deals with the presence of (non-smooth) creases. Although the present thesis deals only with smooth surfaces, it would be interesting to extend to creased surfaces. This is left as future work.

The paper [25] is perhaps the closest to what is done in chapter 5. The authors of the paper suggest a discrete parameterization of the surface rulings, called guiding rules, using the boundary contour. These guiding rules are later smoothed by interpolation using cubic Hermite polynomials. To obtain the rulings, the authors rely on a 3D reconstruction of the embedded surface being available. As will be discussed in chapter 5, the authors also mention the subtle differences between developable surfaces and torsal ruled surfaces which they deal with in the later paper [26] by segmenting into deformation regions.

In [27] the authors propose to reconstruct a 3D surface from a perspective camera with known intrinsic parameters when a template is available and point-wise correspondence is provided. It is based solely on distance constraints so it does not require a smooth embedding of the surface.

Another approach is to describe the surface as an inextensible triangle mesh by imposing distance constraints [34, 35, 33]. With this formulation the authors are able to reconstruct smooth and sharply folded surfaces by relaxing the distance constraints and allowing them to shorten. The authors claim that this is a more faithful representation since extrinsic distance is allowed to shorten, and at the same time obtain a convex optimization problem. The formulation supposes that the correspondence between 3D surface points and 2D locations in the input image is known. In [36] a closed form solution for reconstruction of surfaces by matching individual images to a reference configuration is described.

In [37] the authors reconstructs the triangle mesh without knowing the template of the surface, which is also a property of what's presented in this thesis. Another interesting paper from this research group is [34]. In [29] and [30] the authors are interested in real-time detection and augmentation of deformable surfaces by robustly minimizing an energy cost function.

Finally, a very interesting paper is [14] in which a developable surface (called applicable surface in the paper) is described as a differential equation. The authors show that the information on the bounding contour is sufficient to determine the structure.

The author of this thesis has also published [8] and [9] which are completely described in this thesis and further refined.

1.3 Contribution

This thesis explores the three problems posed at the end of section 1.1, providing a solution and discussing each of them. The exact contributions by chapter are

- Chapter 2 is meant to smooth the transition to the following chapters and contains the following novel ideas
  - A general description on how to compute local factorizations applied to rigid scenes.
  - An algorithm for neighbor identification from scale orthographic images in the context of isometric deformations.
- Chapter 3 describes the surface unfolding problem and a means of obtaining the surface points $\mathcal{Q}$ from the observed 2D points $\mathcal{P}^k$ is described.
- The solution for the pose estimation problem is provided in chapter 4.
- A new (differential) parameterization of developable surfaces is described in chapter 5 which unites generation of the 3D surface and its unfolding.

The following are additional contributions which do not fit in a particular problem:

- An in-depth characterization of a needed set of matrices denoted as "sub-Stiefel is provided in appendix A.
- An algorithm for computing the sub-Stiefel Procrustes solution as described in appendix C.

Chapter 2

Local v.s. Global Factorization

Contents

The previous chapter defined several point sets that will be used throughout this document: $\mathcal{Q} \subset \mathbb{R}^2$, $\mathcal{R}^k \subset \mathbb{R}^3$ and $\mathcal{P}^k \subset \mathbb{R}^2$, where the superscript k denotes the kth input image. It was assumed that all these point sets were commonly numbered, implying that a point matching algorithm was run prior to the discussion. Feature extraction and matching is by itself a subject worthy of a lot of investigation and is out of the scope of this thesis.

Assumption 2.1 (Matched Features Assumption) *The input data consists of feature points and it is assumed that the correspondence between points is available. So if $\mathcal{A}$ and $\mathcal{B}$ are two point sets whose elements are refered to by an indexing function $\mathcal{A} = \{\mathbf{a}_i\}_1^N$ and $\mathcal{B} = \{\mathbf{b}_i\}_1^N$, it is assumed that each point $\mathbf{a}_i$ is related to the corresponding element $\mathbf{b}_i$.*

It is not assumed that feature points are visible in all images.

The notion of **neighborhood** will be very important throughout this document hence some particular notation is introduced. A neighborhood around a given point $\mathbf{q}_i \in \mathcal{Q}$ consists of all the points

$\mathbf{q}_j$ which are somehow considered to be close to $\mathbf{q}_i$. Define the set $\mathcal{N}_i = \{j : \mathbf{q}_j \text{ is a neighbor of } \mathbf{q}_i\}$. The constants N_i will denote the number of elements in the set $\mathcal{N}_i$. Introducing an indexing function for the set $\mathcal{N}_i$, the notation 1_i will be used to refer to the first element of the set, 2_i refers to the second, up to N_{ii} which is the last element. How the set $\mathcal{N}_i$ is chosen is discussed later, for now just consider it as the set of points close to $\mathbf{q}_i$.

Due to the matched features assumption these neighborhoods can be naturally propagated to all other sets $\mathcal{R}^k$ and $\mathcal{P}^k$. Although point $\mathbf{q}_i$ is always a neighbor of point $\mathbf{q}_{j_i}$, there is no guarantee that in a particular image either point is observed so some additional care is needed when stating "$\mathbf{p}_i^k$ is a neighbor of point $\mathbf{p}_{j_i}^k$". Since it will be usual to cycle through all the observed neighborhoods, the set $\mathcal{V} = \{(i,j,k) : \mathbf{p}_i^k \text{ and } \mathbf{p}_{j_i}^k \text{ are visible}\}$ is here defined. This set is the disjoint union of smaller per-image sets $\mathcal{V}^k = \{(i,j) : \mathbf{p}_i^k \text{ and } \mathbf{p}_{j_i}^k \text{ are visible}\}$.

Assumption 2.2 (Scale Orthographic Cameras) *The assumed camera model consists of a scaled projection on a plane. Hence a camera is defined by a positive scale factor $s \in \mathbb{R}^+$ and a Stiefel matrix $\mathbf{P} \in \mathsf{O}(2,3)$, yielding the projection function*

$$C : \mathbb{R}^3 \to \mathbb{R}^2 \quad C(\mathbf{x}) = s\mathbf{P}\mathbf{x}.$$

*The camera parameters are **not** assumed to be known.*

The following chapters will look at equation (1.1) from different perspectives and attempt to solve for unknown variables when only some of them are observed. For convenience, the equation is here repeated and the intermediate 3D points $\mathbf{r}_i^k \in \mathcal{R}^k \subset \mathcal{S}^k$ are evidenced

$$\mathbf{p}_i^k = C^k\left(\mathcal{I}^k\left(\mathbf{q}_i\right)\right) = \begin{cases} \mathbf{r}_i^k = \mathcal{I}^k\left(\mathbf{q}_i\right) \\ \mathbf{p}_i^k = C^k\left(\mathbf{r}_i^k\right) \end{cases} \qquad \forall i,k\,. \tag{2.1}$$

In chapter 1 it was mentioned that the considered embedding functions $\mathcal{I}^k : \mathbb{R}^2 \to \mathbb{R}^3$ had to obey certain metric constraints so as not to stretch or shear the embedded points. Here these embedding functions are defined as functions which do not change lengths of curves on the surface and called **isometries**. So if $c : \mathcal{A} \subset \mathbb{R} \to \mathbb{R}^2$ is any 2D curve of finite length, then the 3D curve $\mathcal{I}^k \circ c$ must have the same length (see theorem 3.10 of [16] for a formal proof or [3, 17] for a general overview of Riemannian geometry). Furthermore, these isometric embedding functions are assumed to be smooth, i.e. infinitely differentiable.

Since the camera model is assumed to be scale orthographic, the camera projection functions C^k

are simple linear transformations as in assumption 2.2. Unfortunately, the isometry functions $\mathcal{I}^k$ do not admit such a simple characterization. This motivates the search for alternative methods of solving equation (2.1).

In order to proceed, a property of isometry functions must be stated:

Fact 2.3 *The Jacobian matrix of an isometry function $\mathcal{I} : \mathbb{R}^2 \to \mathbb{R}^3$ evaluated at any point $\mathbf{q}$ is $\mathbf{J}_{\mathbf{q}} \in \mathsf{O}(3,2)$. Note that here the Stiefel matrix is a 3×3 rotation matrix without the last column.*

Proof This is a simple consequence of an isometry having to preserve the norm of tangent vectors and angles between them. See [3, 17] for details. ∎

So although the isometry functions are hard to represent, their Jacobian matrices have a simple representation. The question becomes how can this be exploited, while still claiming to solve something related to equation (2.1)?

It turns out that differential geometry provides the answer with the notion of tangent vector and push-forwards of functions. Using these notions equation (2.1) also defines a linear relation between the tangent vectors at corresponding points (see for example section IV.1 of [3]). When written in coordinates, this linear relation is commonly known as the Jacobian matrix.

If the Jacobian matrix of the camera projection function C^k at a point $\mathbf{r}_i^k$ is denoted by $\mathbf{C}_i^k \in \mathsf{M}^{2\times 3}$, where $\mathsf{M}^{m\times n}$ is the set of $m \times n$ matrix, equation (2.1) may be rewritten locally in terms of tangent vectors at a point $\mathbf{q}_i$ as

$$\mathbf{v}_i^k = \mathbf{C}_i^k \mathbf{J}_i^k \mathbf{w}_i = \begin{cases} \mathbf{u}_i^k = \mathbf{J}_i^k \mathbf{w}_i \\ \mathbf{v}_i^k = \mathbf{C}_i^k \mathbf{u}_i^k \end{cases} \qquad \forall i, k. \tag{2.2}$$

where $\mathbf{w}_i \in \mathbb{T}_{\mathbf{q}_i}\mathbb{R}^2$, $\mathbf{u}_i^k \in \mathbb{T}_{\mathbf{r}_i^k}\mathcal{S}^k$ and $\mathbf{v}_i^k \in \mathbb{T}_{\mathbf{p}_i^k}\mathbb{R}^2$ are tangent vectors. Since the considered camera projection function is linear, its Jacobian matrix is the linear transformation $\mathbf{C}_i^k = s^k\mathbf{P}^k$, where $s^k \in \mathbb{R}^+$ and $\mathbf{P}^k \in \mathsf{O}(3,2)$ are the camera parameters as defined in the scale orthographic cameras assumption. Note that the Jacobian matrix for this camera projection model does not depend on the point. Substituting in the previous equation results in

$$\mathbf{v}_i^k = s^k\mathbf{P}^k \mathbf{J}_i^k \mathbf{w}_i = \begin{cases} \mathbf{u}_i^k = \mathbf{J}_i^k \mathbf{w}_i \\ \mathbf{v}_i^k = s^k\mathbf{P}^k \mathbf{u}_i^k \end{cases} \qquad \forall i, k. \tag{2.3}$$

This equation is valid when true tangent vectors are considered. Unfortunately, due to the deformation of the embedded surface $\mathcal{S}^k$ the vectors $\mathbf{u}_i^k \in \mathbb{T}_{\mathbf{r}_i^k}\mathcal{S}^k$ cannot be represented by point subtraction

which is only valid on affine spaces. This in turn implies that the true tangent vectors $\mathbf{v}_i^k$ cannot be simply computed as differences of points either. On the other hand, this does not apply to $\mathbf{w}_i$ which are actual tangent vectors to $\mathbb{R}^2$.

2.1 The Planar Case

Consider first the case where all the embedding functions $\mathcal{I}^k$ are rigid transformations, mapping the set $\mathcal{Q} \subset \mathcal{U} \subset \mathbb{R}^2$ to the planes $\mathcal{S}^k \subset \mathbb{R}^3$. In this case the isometry functions are affine functions defined by a Stiefel matrix $\mathbf{J}^k \in \mathsf{O}(3,2)$ and a translation vector $\mathbf{t}^k$, which do not change from point to point.

$$\mathcal{I}^k(\mathbf{q}) = \mathbf{J}^k\mathbf{q} + \mathbf{t}^k \tag{2.4}$$

More important than the structure of $\mathcal{I}^k$, is the fact that their images $\mathcal{S}^k = \mathcal{I}^k(\mathcal{Q})$ are planes isometric to $\mathbb{R}^2$. In this particular case all the vectors in equation (2.3) can be represented as differences of points. So, given two indexes i and j, the following are valid representations for the tangent vectors at each surface

$$\mathbf{w}_i = \mathbf{q}_j - \mathbf{q}_i \qquad \mathbf{v}_i^k = \mathbf{p}_j^k - \mathbf{p}_i^k \qquad \mathbf{u}_i^k = \mathbf{r}_j^k - \mathbf{r}_i^k \tag{2.5}$$

all obeying equation (2.3), so it is perfectly valid to write

$$\mathbf{p}_j^k - \mathbf{p}_i^k = s^k\mathbf{P}^k\mathbf{J}^k\left(\mathbf{q}_j - \mathbf{q}_i\right) \quad \forall (i,j,k) \in \mathcal{V} \tag{2.6}$$

As in any regression problem, in the presence of noise these equations are no longer strictly obeyed and it is usual to find the values that best approximate them in some sense.

2.2 The Non-Planar Case

When $\mathcal{S}^k$ are not planar, equations (2.5) can no longer be written exactly, but in order to proceed they will be approximated by a difference of points as if they belonged to an affine space. This leads to the following approximation

Approximation 2.4 (Locally Planar Approximation) *The surface sampling is dense enough for local neighborhoods in the embedded surfaces to be well approximated by planes.*

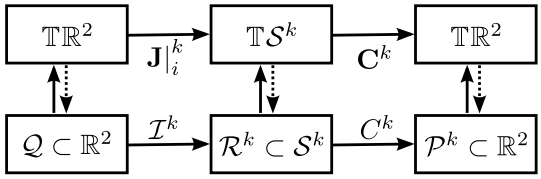


Figure 2.1: Going to the tangent space and back.

which allows for equations similar to (2.5) to be written

$$\mathbf{w}_i = \mathbf{q}_{j_i} - \mathbf{q}_i \qquad \mathbf{v}_i^k \approx \mathbf{p}_{j_i}^k - \mathbf{p}_i^k \qquad \mathbf{u}_i^k \approx \mathbf{r}_{j_i}^k - \mathbf{r}_i^k \tag{2.7}$$

Notice that here the neighboring points notation j_i is used, so the equations are only valid when considering neighboring points. Similarly, equation (2.3) can be approximated as

$$\mathbf{p}_{j_i}^k - \mathbf{p}_i^k \approx s^k \mathbf{P}^k \mathbf{J}_i^k (\mathbf{q}_{j_i} - \mathbf{q}_i) \quad \forall (i,j,k) \in \mathcal{V}. \tag{2.8}$$

In this case, the effects of surface distortion is fused with the effects of noise in the observed samples.

2.3 What is Lost

This chapter suggests that instead of directly solving the equation system defined by (2.1) it is easier to solve the equation system defined by equations (2.3) instead. Is there any relevant information not captured by the suggested alternative? The horizontal lines in figure 2.1 graphically represent equations (2.1) and (2.3). The vertical up arrows represents equations (2.5). The question posed here is if anything relevant is lost when coming back down through the dotted vertical arrows. The process of coming down will be called **integrating** the tangent data and the answer to the question is that not much is lost, but this needs to be clarified.

To integrate the tangent vectors back to the surface, a function known as the **exponential function** needs to be available, which describes how to travel from a point along a tangent direction. In the planar case, this function is simply

$$\exp_{\mathbf{x}}(\mathbf{v}_{\mathbf{x}}) = \mathbf{x} + \mathbf{v}_{\mathbf{x}} \quad \forall \mathbf{x} \in \mathbb{R}^n, \mathbf{v}_{\mathbf{x}} \in \mathbb{T}_{\mathbf{x}}\mathbb{R}^n$$

which is a fancier way of writing equations (2.5). In the noiseless case, this allows integration to

be performed by starting from a point, building all it's neighbors using the exponential function and applying it recursively starting from the neighboring points. The only ambiguity is the starting point, this information is lost when going to the tangent space and cannot be recovered when integrating. This is not considered to be a loss since this information is actually not obtainable from the observed data.

The exact same argument applies to the non-planar case. The added difficulty in this case is that the exponential function is usually unknown and equation (2.5) is used as an approximation.

When the existence of noise is considered in the problem a new issue arises: noise integration. In the simplistic algorithm for integration just given, the problem occurs when one builds a neighbor with noise and this noise is propagated to all points reconstructed from that point onward. The algorithm for integration will not be as simplistic as the one just described, but this issue still remains and can be more or less relevant depending on the neighbor graph connectivity and noise distribution. If the noise has zero mean, then simply augmenting the number of neighbors helps to contain this effect. Unfortunately in the non-planar case, where the surface bend is treated as noise, the "noise" distribution cannot be assumed to have zero-mean. The next section looks at the effect of noise in a little more detail.

2.4 Local Factorization Method

Section 2.2 suggests using a local reconstruction method. To gain a little insight and ease the transition to the next chapter this section will describe how to implement a local factorization algorithm and use it in the same context as the Tomasi-Kanade algorithm which uses all points at once. Note that the algorithm presented in this section is purely for discussion purposes but will provide a high level of parallelism with the algorithm that will be presented in the next section.

2.4.1 Neighbor Estimation

The notion of locality has been presented but no hint on how to find the local neighbors has been given. This problem is simple when the points $\mathbf{q}_i$ are available, but it is not trivial when only the camera projected points $\mathbf{p}_i^k$ are known since the actual distances between them is not known. The problem is aggravated by allowing the surface to fold, which allows intrinsically far points to appear closer than the actual neighbors, and further by the fact that scaled-orthographic cameras are considered. Although very simple, one can consider an algorithm that classifies pairs of points as neighbors if and only if they are "close" in every image where the pair is visible. The algorithm, described in table 2.1

Neighbor Estimation

Input: Matched point clouds $\mathcal{P}^k$. Minimum number of images L where each point is observed and the desired number of neighbors N for each point. Any point that is seen in less than L images is completely discarded.

Output: Builds the sets $\mathcal{N}_i$ representing the neighboring points of point i.

1: For each point i and image k where point i is visible, compute the distance d_{ij}^k, in image coordinates, to every other visible point. Any point that is not visible in at least L images simultaneously with point i is not considered a candidate as a neighbor of point i.

2: Sort the distances in each image ascendantly and attribute a ranking r_{ij}^k according to the sorted position (so $r_{ij}^k = 1$ means that point j is the closest to point i in image k, $r_{ij}^k = 2$ is the second closest, etc)

3: Let $r_{ij} = \max_k r_{ij}^k$. This is the worst rank of each neighbor in all images where i and j are observed simultaneously.

4: For each point i, pick the N points with lowest worst rank r_{ij}. These are the elements of the output set $\mathcal{N}_i$. Ties between 2 points can be broken by contemplating relative distances. Let $\mathcal{D}$ be the index of the images where i, j_1 and j_2 are simultaneously visible, choose j_1 if $\prod_{k\in\mathcal{D}} d_{ij_1}^k < \prod_{k\in\mathcal{D}} d_{ij_2}^k$ otherwise pick j_2. Ties between a higher number of points can be broken similarly by stating that $\mathcal{D}$ is the set of images where all points are seen. If $\mathcal{D}$ is empty, break ties arbitrarily.

Table 2.1: Neighbor Estimation Algorithm.

is applied to each point independently and needs two parameters besides the observed point clouds: L is the minimum number of images in which a point pair needs to be visible to be accepted, and N is the desired number of neighbors for each point. It works by excluding non-neighbors, but does not guarantee that all returned points are actual neighbors. In practice, as long as the images taken from general enough configurations, the algorithm has provided very good results for the considered cases.

Note that the algorithm described produces good results for the purposes of this document but there is margin for improvement. For example, step 2 should still allow for many more than N points to be available for neighbor selection, or else risk a point being considered a neighbor simply because all others where occluded in too many frames. In all the experiments in this thesis, it is considered that points are visible in "enough" images and this point will not be discussed further. If in some application this is not so simple, care must be taken as to how the problem will be solved, e.g. take more images, relax L, allow a varying number of neighbors N_i, etc.

The neighbor detection step may also be used to detect false matches between images. Suppose a fairly large number of images are available and that a pair of points are considered neighbors in all but a few images. Then some feedback to the matching algorithm might be provided for it to review the

quality of the matching step in these images. In this dissertation, the matching of points is assumed perfect (assumption 2.1) and no further discussion on this topic is given.

2.4.2 Local Factorization

Consider in this section that one is interested in solving the system of equations (as in equations (1.2))

$$\mathbf{p}_i^k = \mathbf{C}^k \mathbf{r}_i \quad \forall i, k. \tag{2.9}$$

where $\mathbf{C}^k \in \mathsf{O}(2,3)$ when an orthographic camera is used, and the 3D points do not depend on the image k. Written locally, this equation becomes

$$\mathbf{v}_i^k = \mathbf{C}^k \mathbf{u}_i \quad \forall i, k. \tag{2.10}$$

and the vectors are defined by a difference of local neighbors:

$$\mathbf{p}_{j_i}^k - \mathbf{p}_i^k = \mathbf{C}^k (\mathbf{r}_{j_i} - \mathbf{r}_i) \quad \forall (i,j,k) \in \mathcal{V}. \tag{2.11}$$

Assuming that all points are seen in every image, collect the previous equations in a matrix form as

$$\mathbf{V}_i = \mathbf{C}\,\mathbf{U}_i \tag{2.12}$$

where

$$\mathbf{V}_i = \begin{bmatrix} \mathbf{p}_{1_i}^1 - \mathbf{p}_i^1 & \mathbf{p}_{2_i}^1 - \mathbf{p}_i^1 & \cdots & \mathbf{p}_{N_{ii}}^1 - \mathbf{p}_i^1 \\ \mathbf{p}_{1_i}^2 - \mathbf{p}_i^2 & \mathbf{p}_{2_i}^2 - \mathbf{p}_i^2 & \cdots & \mathbf{p}_{N_{ii}}^2 - \mathbf{p}_i^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_{1_i}^K - \mathbf{p}_i^K & \mathbf{p}_{2_i}^K - \mathbf{p}_i^K & \cdots & \mathbf{p}_{N_{ii}}^K - \mathbf{p}_i^K \end{bmatrix} \qquad \mathbf{C} = \begin{bmatrix} \mathbf{C}^1 \\ \mathbf{C}^2 \\ \vdots \\ \mathbf{C}^K \end{bmatrix} \tag{2.13}$$

$$\mathbf{U}_i = \begin{bmatrix} \mathbf{r}_{1_i} - \mathbf{r}_i & \mathbf{r}_{2_i} - \mathbf{r}_i & \cdots & \mathbf{r}_{N_{ii}} - \mathbf{r}_i \end{bmatrix} \tag{2.14}$$

Applying the rank factorization algorithm, the objective is to infer the values of $\mathbf{C}$ and $\mathbf{U}_i$ from the observed values $\mathbf{V}_i$. Note that the rank factorization algorithm is being applied locally to each neighborhood at each point. In this section a rigid 3D scene is considered so a rank 3 factorization is applied to each matrix $\mathbf{V}_i$ resulting in

$$\mathbf{V}_i \approx \hat{\mathbf{C}}_i \hat{\mathbf{U}}_i \tag{2.15}$$

The solution is not unique, and any invertible matrix $\mathbf{G}_i \in \mathbb{GL}(3)$ generates a new solution $\hat{\mathbf{C}}_i \mathbf{G}_i^{-1}$ and $\mathbf{G}_i \hat{\mathbf{U}}_i$. These matrices will be used to integrate the solution from vectors back into points.

Notice that in the standard factorization problem there is only a single $\mathbf{C}$ which accounts for all the points, while here a $\mathbf{C}_i$ matrix is computed at each point containing information about its neighbors. Although in this toy problem the advantage of doing this is not clear, this is done so as to be closer to the later employed method to reconstruct embedded surfaces.

2.4.3 Integrating the Solution

Recovering the points from the vector information is accomplished by solving

$$\begin{bmatrix} \mathbf{r}_{1_i} - \mathbf{r}_i & \mathbf{r}_{2_i} - \mathbf{r}_i & \dots & \mathbf{r}_{N_{ii}} - \mathbf{r}_i \end{bmatrix} = \mathbf{G}_i \hat{\mathbf{U}}_i \quad \forall i \tag{2.16}$$

where $\hat{\mathbf{U}}_i$ is the solution found previously and everything else is a variable of the problem. Notice that now one is not considering a single neighborhood, but all at once. Collecting all the variables into a matrix $\mathbf{X}_i \in \mathsf{M}^{3\times 4N}$:

$$\mathbf{X} = \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \dots & \mathbf{r}_N & \mathbf{G}_1 & \mathbf{G}_2 & \dots & \mathbf{G}_N \end{bmatrix}$$

allows all the equations described by (2.16) to be written in matrix form as

$$\mathbf{X}\mathbf{A} = \mathbf{0}$$

for some matrix $\mathbf{A}$ defined by the linear equations (2.16). This equation is left invariant in the sense that if $\hat{\mathbf{X}}$ is a solution then left multiplication by any matrix $\mathbf{H} \in \mathsf{M}^{3\times 3}$ originates an equally valid solution $\mathbf{H}\hat{\mathbf{X}}$. This allows to search for $\mathbf{X}$ only in the Stiefel matrices since any matrix can be decomposed as a Stiefel matrix and a square matrix (known as the polar decomposition in theorem 7.3.2 of [15]). Also the structure of the problem forces matrix $\mathbf{A}$ to have a trivial kernel when all the points $\mathbf{r}_i$ are equal and $\mathbf{G}_i = \mathbf{0}$. This solution holds no information and needs to be discarded. When no noise is present the true points $\mathbf{r}_i$ and the true matrices $\mathbf{G}_i$ also obey the equation, meaning that matrix $\mathbf{A}$ has kernel dimension at least 4 (three from the true $\mathbf{X}$ and trivial solution mentioned above). Unfortunately, proving that the matrix has kernel dimension exactly 4, depends on the neighbor graph connectivity and will not be done. Numerically it is easy to convince that if the neighboring points are not singularly distributed and the neighboring graph is connected, then this matrix will have rank 4.

With this in mind, an optimization problem can be formulated as finding the 3 dimensional subspace with the lowest residual, orthogonal to the trivial solution $\mathbf{z} = \begin{bmatrix}\mathbf{1}_N^T & \mathbf{0}_{3N}^T\end{bmatrix}^T$:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_i \langle \mathbf{XA}, \mathbf{XA} \rangle \\ \text{s.t.} \quad & \mathbf{X} \in \mathsf{O}(3, 4N) \\ & \mathbf{Xz} = \mathbf{0} \end{aligned} \tag{2.17}$$

where here $\mathbf{X}$ is seen as a projector of $\mathbf{A}$ to a lower 3-dimensional space. Expanding the inner product in equation (2.17) this can be re-written as

$$\begin{aligned} \text{minimize} \quad & \operatorname{tr}\left\{\mathbf{XAA}^T\mathbf{X}^T\right\} \\ \text{s.t.} \quad & \mathbf{X} \in \mathsf{O}(3, 4N) \\ & \mathbf{Xz} = \mathbf{0} \end{aligned} \tag{2.18}$$

Which, accordingly to the Rayleigh-Ritz quotient (see for example theorem 4.2.2 in [15]), is exactly the formulation of computing the second and third lowest singular values of a symmetric matrix $\mathbf{AA}^T$ where the least singular vector is known to be $\begin{bmatrix}\mathbf{1}_N^T & \mathbf{0}_{3N}^T\end{bmatrix}^T$. This can be obtained with available software taking into account the sparsity of matrix $\mathbf{A}$ such as function `svds` included in MATLAB or see for example [1].

After a solution $\hat{\mathbf{X}}$ is obtained, it contains all the reconstructed points and matrices $\mathbf{G}_i$ up to a global multiplication matrix $\mathbf{H}$. So the solution to the original factorization problem (2.15) is updated with the computed $\mathbf{G}_i$ and changed to

$$\begin{aligned} \hat{\mathbf{C}}_i &\leftarrow \hat{\mathbf{C}}_i \mathbf{G}_i^{-1} \\ \hat{\mathbf{U}}_i &\leftarrow \mathbf{G}_i \hat{\mathbf{U}}_i. \end{aligned}$$

In the no-noise case all $\hat{\mathbf{C}}_i$ should be equal. When noise is considered and the reconstruction process is working correctly, they should be similar.

2.4.4 Forcing the Stiefel Constraints

The last step of the classic Tomasi-Kanade algorithm approximates all the $\mathbf{C}^k$ matrices to the Stiefel set using a least squares error function. Here the exact same step is taken to approximate all the 2×3

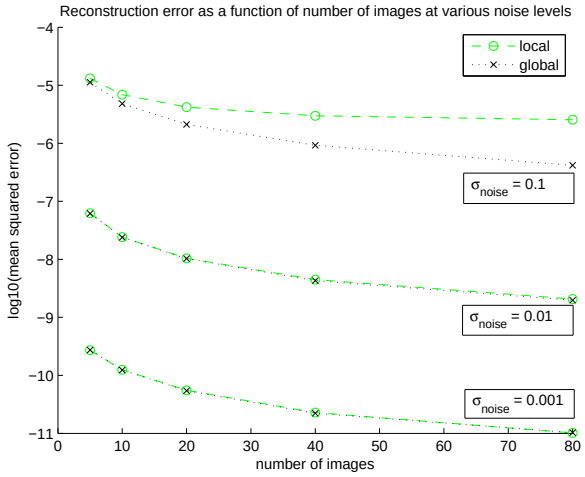


Figure 2.2: Reconstruction error as a function of number of images at different noise levels (standard deviation of 0.001, 0.01 and 0.1). The other parameters are constant, with 200 points and 20 neighbors.

sub-matrices of $\hat{\mathbf{C}}_i$:

$$\hat{\mathbf{C}}_i = \begin{bmatrix} \hat{\mathbf{C}}_i^1 \\ \hat{\mathbf{C}}_i^2 \\ \vdots \\ \hat{\mathbf{C}}_i^K \end{bmatrix}$$

using the fact that a single matrix $\mathbf{H}$ can be multiplied to the obtained $\mathbf{G}_i$ without changing the optimality of the previous steps.

2.4.5 Results

To test the local algorithm introduced in this section against the global Tomasi Kanade algorithm, random points were generated using a multivariate Gaussian distribution normalized to have unit variance.

In the noiseless case the two methods produce the same result, aside from a global rotation and translation, which are intrinsically ambiguous in the problem. As the results in figures 2.2, 2.3 and 2.4 show, the local version results are only distinguishable from the global version when the noise level is high. At low noise levels, both algorithms produce comparable results as long as a sufficient number of neighbors are used. As the noise level increases, the number of neighbors required for reconstruction also increases.

In figures 2.2, 2.4 notice that at the highest noise level ($\sigma_{\text{noise}} = 0.1$), the local version seems to level off instead of tending asymptotically to zero. This is due to the fact that noise has a much higher

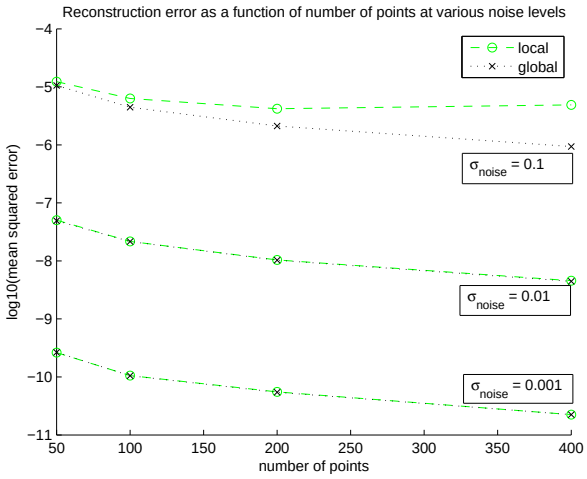


Figure 2.3: Reconstruction error as a function of number of points at different noise levels (standard deviation of 0.001, 0.01 and 0.1). The other parameters are constant, with 20 images and 20 neighbors.

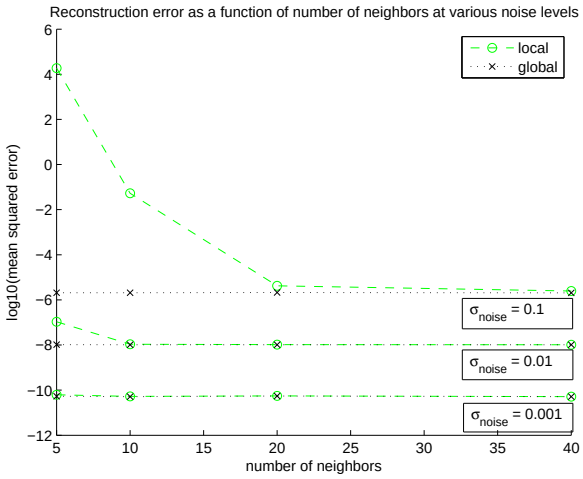


Figure 2.4: Reconstruction error as a function of number of neighbors at different noise levels (standard deviation of 0.001, 0.01 and 0.1). The other parameters are constant, with 20 images and 200 points. Note that the global algorithm does not use neighbors, hence it is constant and serves only for comparison.

impact when subtracting points close together than points far away (relative noise). As points are randomly added according to a Gaussian distribution, neighborhoods become closer together which increases relative noise.

In terms of complexity, the local method is more complex and its use in this problem is not justifiable. As the next chapters show, the local method can be used to solve a broader class of problems to which the direct Tomasi-Kanade algorithm is not applicable.

Chapter 3

The Surface Unfolding Problem

Contents

Starting from several observations of the feature points $\mathcal{P}^k$, the problem described in this chapter consists of finding the flat surface points $\mathcal{Q}$ that generated them through equation (1.1), here repeated for convenience

$$\mathbf{p}_i^k = C^k\left(\mathcal{I}^k\left(\mathbf{q}_i\right)\right) \quad \forall i,k.$$

The left side of the equation consists of observed image points. In the **surface unfolding** problem everything else is unknown aside from the assumption that the cameras are scale-orthographic as in assumption 2.2 and the first order characterization of the isometric embedding functions as in fact 2.3. This leads to the problem statement

Problem Statement 3.1 (Surface Unfolding Problem) *Starting with sets $\mathcal{P}^k \subset \mathbb{R}^2$ of image point observations known to have been generated by the model*

$$\mathbf{p}_i^k = C^k\left(\mathcal{I}^k\left(\mathbf{q}_i\right)\right) \quad \forall i,k. \tag{3.1}$$

recover the generating set $\mathcal{Q} \subset \mathbb{R}^2$ under the following conditions:

- *C^k represents the projection model of a scale-orthographic camera (see assumption 2.2).*
- *$\mathcal{I}^k : \mathbb{R}^2 \to \mathbb{R}^3$ are isometric embedding functions.*
- *All points have been previously matched and the correspondence between points in different $\mathcal{P}^k$ is known (see assumption 2.1 and the discussion immediately after).*

Chapter 2 argued that instead of solving the generating equation directly, one should solve the local version (2.3) instead, which is here repeated for convenience

$$\mathbf{v}_i^k = s^k \mathbf{P}^k \mathbf{J}_i^k \mathbf{w}_i = \begin{cases} \mathbf{u}_i^k = \mathbf{J}_i^k \mathbf{w}_i \\ \mathbf{v}_i^k = s^k \mathbf{P}^k \mathbf{u}_i^k \end{cases} \quad \forall i,k. \tag{3.2}$$

where $\mathbf{w}_i \in \mathbb{T}_{\mathbf{q}_i}\mathbb{R}^2$, $\mathbf{u}_i^k \in \mathbb{T}_{\mathbf{r}_i^k}\mathcal{S}^k$ and $\mathbf{v}_i^k \in \mathbb{T}_{\mathbf{p}_i^k}\mathbb{R}^2$ are tangent vectors. The scale factors s^k and the matrices $\mathbf{P}^k$ are the Jacobian matrices of the scaled orthographic camera model (see assumption 2.2 and the discussion leading to equation (2.3)), and $\mathbf{J}_i^k$ are the Jacobian matrices of the isometry functions $\mathcal{I}^k$ at each point $\mathbf{q}_i$ (see fact 2.3).

The first observation is that camera motion is itself an isometry, hence any set of matrices $\mathbf{R}^k \in \mathsf{O}(3)$ representing a rotation of the camera may be "absorbed" by the isometry parameters in equation (3.2) as

$$\mathbf{v}_i^k = s^k \underbrace{\mathbf{P}^k\mathbf{R}^k}_{\hat{\mathbf{P}}^k} \underbrace{\left(\mathbf{R}^k\right)^T \mathbf{J}_i^k}_{\hat{\mathbf{J}}_i^k} \mathbf{w}_i \qquad \forall i,k.$$

This is possible since the parameters $\mathbf{P}^k$ and $\mathbf{J}_i^k$ are not considered relevant for the solution of problem 3.1. Also the problem formulation does not provide enough constraints to obtain a unique solution for these. Thus the free $\mathbf{R}^k$ can be used to fix $\hat{\mathbf{P}}^k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ which, when multiplied, removes the bottom row of the Jacobian matrices $\hat{\mathbf{J}}_i^k$. The set of all matrices which are the topmost 2×2 block of a 3×2 Stiefel matrix is named in this thesis to be the **sub-Stiefel** set and denoted by $\mathbb{SS}$. These

matrices are described in detail in appendix A, which will be refered to when certain properties of these matrices are needed. For now it is enough to know that using these matrices allows equation (3.2) to be rewritten as

$$\mathbf{v}_i^k = s^k \mathbf{O}_i^k \mathbf{w}_i \qquad \forall i, k.$$

where $\mathbf{O}_i^k \in \mathbb{SS}$. When the locally planar approximation 2.4 is used, this equation can be written in terms of differences of points as

$$\mathbf{p}_{j_i}^k - \mathbf{p}_i^k \approx s^k \mathbf{O}_i^k (\mathbf{q}_{j_i} - \mathbf{q}_i) \quad \forall (i, j, k) \in \mathcal{V}. \tag{3.3}$$

As long as all the points in the neighborhood of $\mathbf{q}_i$ are visible in all images the objects of the equation can be collected into matrices

$$\mathbf{V}_i = \begin{bmatrix} \mathbf{p}_{1_i}^1 - \mathbf{p}_i^1 & \mathbf{p}_{2_i}^1 - \mathbf{p}_i^1 & \dots & \mathbf{p}_{N_{i_i}}^1 - \mathbf{p}_i^1 \\ \mathbf{p}_{1_i}^2 - \mathbf{p}_i^2 & \mathbf{p}_{2_i}^2 - \mathbf{p}_i^2 & \dots & \mathbf{p}_{N_{i_i}}^2 - \mathbf{p}_i^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{p}_{1_i}^K - \mathbf{p}_i^K & \mathbf{p}_{2_i}^K - \mathbf{p}_i^K & \dots & \mathbf{p}_{N_{i_i}}^K - \mathbf{p}_i^K \end{bmatrix} \qquad \mathbf{M}_i = \begin{bmatrix} s^1 \mathbf{O}_i^1 \\ s^2 \mathbf{O}_i^2 \\ \vdots \\ s^K \mathbf{O}_i^K \end{bmatrix}$$

$$\mathbf{W}_i = \begin{bmatrix} \mathbf{q}_{1_i} - \mathbf{q}_i & \mathbf{q}_{2_i} - \mathbf{q}_i & \dots & \mathbf{q}_{N_{i_i}} - \mathbf{q}_i \end{bmatrix}$$

allowing it to be written as

$$\mathbf{V}_i \approx \mathbf{M}_i \mathbf{W}_i. \tag{3.4}$$

Since $\mathbf{V}_i$ are obtained from the observations, the solution to this equation can be found by a factorization algorithm similar to the local factorization algorithm presented earlier. The full problem is broken in much simpler subproblems, chained together to obtain the final result as shown in table 3.1 and represented graphically in figure 3.1

The first step of the algorithm has already been described in section 2.4.1, the others will be treated in the following sections.

Note that while in section 2.4.2, using local neighbors was merely academic and for comparison purposes, here it is a requirement since the Jacobian matrices $\mathbf{J}_i^k$ are changing from point to point.

Surface Unfolding

Input: Matched point clouds $\mathcal{P}^k$. Minimum number of images L where each point is observed and the desired number of neighbors N for each point.

Output: The set $\mathcal{Q}$ of unfolded points, the camera scale factors s^k (up to a global scale factor) and the sub-Stiefel matrices $\mathbf{O}_i^k$.

1: Discover local neighbors from the observed images, that is, for each i, build the index set $\mathcal{N}_i = \{j\}_1^{N_i}$ as in the discussion below assumption 2.1;

2: Use bilinear factorization to factor equation (3.4);

3: Approximate the shape consistency constraints (i.e. integrate the solution) $\mathbf{W}_i \approx \begin{bmatrix}\mathbf{q}_{1_i} - \mathbf{q}_i & \mathbf{q}_{2_i} - \mathbf{q}_i & \dots & \mathbf{q}_{N_{ii}} - \mathbf{q}_i\end{bmatrix}$;

4: Approximate the model consistency constraints $\mathbf{O}_i^k \in \mathbb{SS}$ and make sure that, together with s^k, they approximate the camera model.

Table 3.1: Surface unfolding algorithm.

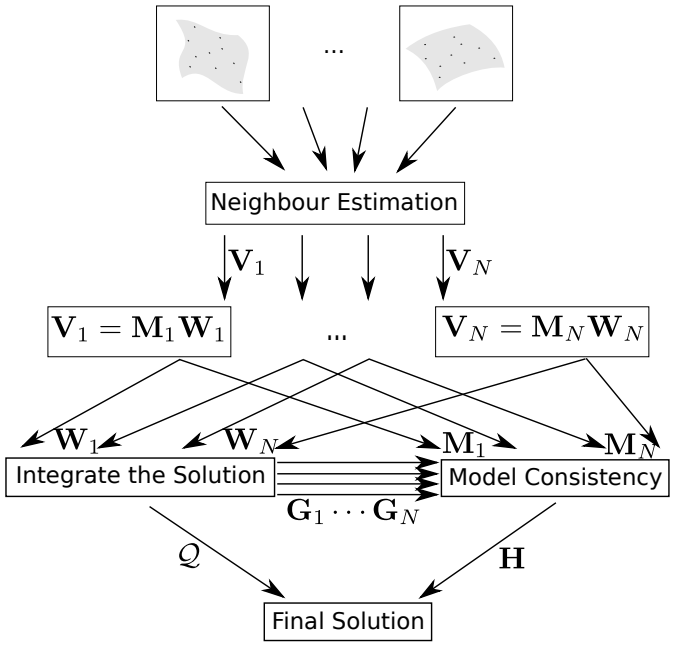


Figure 3.1: Schematic diagram of the local factorization algorithm applied to isometric surfaces.

3.1 Bilinear Factorization

Equation (3.4) states that the observations $\mathbf{V}_i$ must be rank 2, so the first step after neighborhood discovery is to project these observation matrices to the rank 2 set in a least-squares sense. This is accomplished by performing an SVD on $\mathbf{V}_i$ and discarding all but the first 2 singular values and vectors. The SVD provides candidates for matrices $\hat{\mathbf{M}}_i$ and $\hat{\mathbf{W}}_i$, but similarly to what happened in section 2.4.2, the solution is not unique and there are matrices $\mathbf{G}_i \in \mathbb{GL}(2)$ that allow to navigate inside the space of all solutions. The differences with respect to what was discussed in section 2.4.2 are that it is a rank 2 factorization instead of rank 3, and some matrices change names (compare equation (2.15) with equation (3.4)).

When occlusion is considered, matrix $\mathbf{V}_i$ is only partially defined (has missing entries) so a rank completion algorithm must be applied [5, 13]. Rank completion algorithms are usually hand in hand with the factorization method already discussed so using it here is a trivial extension.

3.2 Integrating the Solution

The only difference with respect to what was said in section 2.4.3 is that, since the reconstruction is planar, $\mathbf{X}$ belongs to $\mathsf{O}(2, 3N)$ and the solution must be orthogonal to the vector $\mathbf{z} = \begin{bmatrix}\mathbf{1}_N & \mathbf{0}_{2N}\end{bmatrix}$. Everything else applies verbatim.

After this step is complete, matrices $\hat{\mathbf{M}}_i$ and $\hat{\mathbf{W}}_i$ should be multiplied by the obtained $\hat{\mathbf{G}}_i$ matrices as:

$$\hat{\mathbf{M}}_i \leftarrow \hat{\mathbf{M}}_i \hat{\mathbf{G}}_i^{-1}$$
$$\hat{\mathbf{W}}_i \leftarrow \hat{\mathbf{G}}_i \hat{\mathbf{W}}_i.$$

3.3 Forcing the Sub-Stiefel Constraints

This is the main difference from what was presented in section 2.4. Previously the camera model matrices should approximate the Stiefel matrices as close as possible, here the $\mathbf{M}_i$ must approximate scaled sub-Stiefel matrices, as hinted in the discussion up to equation (3.4).

To force the sub-Stiefel constraints, there is still a global $\mathbf{H} \in \mathbb{GL}(2)$ free matrix that navigates the space of solutions while still maintaining the previously imposed conditions. This matrix, along with the merged scale factors s^k in $\hat{\mathbf{M}}_i$, will be used to "straighten the axes".

3.3.1 Distance Functions

Before continuing with the actual problem details a slight sidetrack is needed. Anticipating the final result, the end result of section 3.3 shall consist of some sort of cost function grossly of the form

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i,k} d^2_{\mathbb{SS}} \left(\hat{\mathbf{M}}_i^k \, \mathbf{H} / s^k \right) \\ \text{s.t.} \quad & \mathbf{H} \in \mathbb{GL}(2) \\ & s^k \in \mathbb{R}^+ \end{aligned} \tag{3.5}$$

This is not the actual cost function that will be used, just a coarse idea that is easily grasped at this point. Basically, the problem is to search for a $\mathbf{H} \in \mathbb{GL}(2)$ matrix and per image scale factors s^k that forces the matrices $\mathbf{O}_i^k = \hat{\mathbf{M}}_i^k \mathbf{H} / s^k$ to minimize some sort of distance to the Sub-Stiefel matrix set.

In the appendix it is proven that the set of sub-Stiefel matrices is equal to the set of 2×2 matrices whose largest singular value is 1 (see theorem A.5). This hints that the distance function $d_{\mathbb{SS}}(\cdot)$ should depend on the maximum singular value of its argument, but other than this and the fact that it should somehow measure a distance to the sub-Stiefel set, there is no naturally given choice of function. This section's objective is to give a convincing argument that there's one particular good choice.

Let's consider a grossly different function, with given scalar values $m_i \in \mathbb{R}$ instead of matrices and using an additive perturbation instead of multiplicative. The objective is to center a point cloud $\{m_i\}_i^N \subset \mathbb{R}$ around a given value a:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_i (m_i + h - a)^2 \\ \text{s.t.} \quad & h \in \mathbb{R} \end{aligned}$$

It is clear that for this cost function the solution is $h^* = -\mathbb{E}[m_i] + a$ where $\mathbb{E}$ denotes the mean value over i function. This solution has the additional property that $\mathbb{E}\left[m_i + h^*\right] = a$ thus confirming that in a certain sense, the translated cloud is indeed centered around a as intended.

Consider now a different problem with all $m_i > 0$ affected by a multiplicative perturbation. While the previous distance function was $d(a,b) = |a - b|$, now a new one is used $d(a,b) = |\log(a/b)|$:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_i \log^2 \left(\frac{m_i h}{a} \right) \\ \text{s.t.} \quad & h \in \mathbb{R}^+ \end{aligned} \tag{3.6}$$

The solution to this problem is also simple once one realizes that it can be converted to the previous problem by changing all variables with their $\log$. Again, the solution will have the same "center-

ing" property, only now in a multiplicative sense due to the $\log$ change of variables: $\mathbb{E}\left[\log(m_i h^*)\right] = \log(a)$. Removing the $\log$, this means that a is the geometric mean of $m_i h^*$ $\left(\text{i.e. } a = \prod_{i=1}^{N} (m_i h^*)^{1/N}\right)$.

Since no natural choice of distance function exists to use in problem (3.5), the best that can be done to narrow the choice is impose desired properties for the solution. A natural one is this centering property just discussed. With this in mind and the characterization of the Sub-Stiefel matrix set given in theorem A.5, the following optimization problem is proposed:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i,k} \log^2 \left(\sigma_{\max}\left(\mathbf{M}_i^k \, \mathbf{H}/s^k\right)\right) \\ \text{s.t.} \quad & \mathbf{H} \in \mathbb{GL}(2) \\ & s^k \in \mathbb{R}^+ \end{aligned} \tag{3.7}$$

where $\sigma_{\max}(\cdot)$ returns the maximum singular value. The cost function is essentially the same as the previous one, using matrices instead, and it measures the distance of the maximum singular value of the argument to $a = 1$. In appendix B it is proven that the solution $\hat{\mathbf{H}}$ and $\hat{s}^k$ verify the condition $\prod_{i=1}^{N} \sigma_{\max}(\mathbf{M}_i^k \, \hat{\mathbf{H}}/\hat{s}^k)^{1/N} = 1$. No equivalent property would exist if the usual squared norm distance function $d(\sigma_{\max}(\mathbf{S}_i \, \mathbf{H}^*), 1) = |\sigma_{\max}(\mathbf{S}_i \, \mathbf{H}^*) - 1|$ were used. Although not canonically chosen, the centering property makes this cost function a nicer candidate over the absolute value of the difference.

The details on how to solve the optimization problem in equation (3.7) are left to appendix B. The main points that should be considered are:

- The optimization problem reduces itself to a compact 2 dimensional optimization problem, which makes it particularly easy to find a local minimum.
- Unfortunately it sometimes exhibits multiple local minimums, but these are so close together that the particular choice is irrelevant in practice.
- It is not smooth everywhere, but the minimum solution does not usually fall on a non-smooth point so it does not hurt gradient based algorithms.

After finding a solution to this problem, the output $\hat{\mathbf{H}}$ changes the previously found values as

$$\begin{aligned} \hat{\mathbf{M}}_i &\leftarrow \hat{\mathbf{M}}_i \hat{\mathbf{H}} \\ \hat{\mathbf{W}}_i &\leftarrow \hat{\mathbf{H}}^{-1} \hat{\mathbf{W}}_i \\ \hat{\mathbf{q}}_i &\leftarrow \hat{\mathbf{H}}^{-1} \hat{\mathbf{q}}_i \end{aligned}$$

and the approximate sub-Stiefel matrices are

$$\hat{\mathbf{O}}_i^k = \hat{\mathbf{M}}_i^k / \hat{s}^k \, .$$

Of these, $\hat{\mathcal{Q}} = \{\hat{\mathbf{q}}_i\}_1^N$ are the solution to problem 3.1 posed at the beginning of this chapter.

3.4 Improving the Solution

It is hard to characterize the obtained solution in terms of optimality. Although bilinear factorization obtains the optimal rank 2 factorization of $\mathbf{V}_i$ in a least squares sense, i.e it solves the optimization problem

$$\begin{aligned} \text{minimize} \quad & \| \mathbf{V}_i - \mathbf{M}_i \mathbf{W}_i \|_2^2 \quad , \\ \text{s.t.} \quad & \mathbf{M}_i \in \mathsf{M}^{2K_i \times 2} \\ & \mathbf{W}_i \in \mathsf{M}^{2 \times N_i} \end{aligned}$$

all other steps approximate the needed constraints while still maintaining this condition. This leads to the question "what objective function should be optimized"? The original problem statement answers this question with equation (3.1) which when certain measurement noise properties are assumed, suggests the following problem:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i,k)\in\mathcal{V}} \left\| \mathbf{p}_i^k - s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathcal{I}^k(\mathbf{q}_i) \right\|_2^2 \\ \text{s.t.} \quad & s^k \in \mathbb{R}^+ \\ & \mathcal{I}^k \text{ isometry} \\ & \mathbf{q}_i \in \mathbb{R} \quad \forall i \end{aligned} \tag{3.8}$$

This problem is very hard to solve due to the second constraint as previously discussed. A similar, but not equivalent, problem can be written based on equation (3.3) instead:

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i,j,k)\in\mathcal{V}} \left\| \left(\mathbf{p}_{j_i}^k - \mathbf{p}_i^k \right) - s^k \mathbf{O}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i \right) \right\|_2^2 \\ \text{s.t.} \quad & s^k \in \mathbb{R}^+ \quad \forall k \\ & \mathbf{O}_i^k \in \mathbb{SS} \quad \forall i,k \\ & \mathbf{q}_i \in \mathbb{R} \quad \forall i \end{aligned} \tag{3.9}$$

This problem is also hard to solve without a good initial approximate solution due to non-convexity and high dimensionality, but can effectively be used to improve the solution previously obtained with the algorithm described in table 3.1. Here the word "improve" is highly dependent on the assumption that this cost function is better than the solution given by the previous algorithm. The only argument that will be given in this thesis preferring the output of one over the other is the quality of the obtained results when compared with ground truth data. Either way, one must always compute the initial solution using the previous algorithm since the problem in equation (3.9) is only solvable if an initialization is provided.

There are many optimization algorithms that can be used to locally optimize equation (3.9) given an initial starting point. The hardest condition is the sub-Stiefel condition, but it is relatively easy to overcome by either decomposing $\mathbf{O}_i^k$ with an SVD decomposition and using the maximum singular value characterization, or by decomposing it using the result of theorem A.9. Here a different route will be taken using coordinate descent, due to the interesting side-problem that it generates which might be applicable in other contexts. The idea is that if the variable sets $\left\{s^k\right\}_1^K$, $\mathcal{Q}$ and $\{\mathbf{O}_i^k : \forall i,k\}$ are considered separately, the solution to the problem in equation (3.9) is globally solvable. In the variable set $\left\{s^k\right\}_1^K$ the problem reduces itself to a least squares problem (if the sign obtained is ever negative, let the variables $\mathbf{O}_i^k$ absorve it). The same is true for the variable set $\{\mathbf{q}_i\}_1^N$. Unfortunately, in the variable set $\{\mathbf{O}_i^k : \forall i,k\}$ the problem is a little harder, but it has been successfully solved (see appendix C). The solution is closed form up to a dependence on the factorization of a degree 6 polynomial.

At the end of section 3.3 a candidate solution to the surface unfolding problem was obtained. This solution does not force the sub-Stiefel constraint, but rather approximates it. Since these variables are the only ones that do not obey the constraint set of the problem in equation (3.9), this is the obvious starting point. An iterative algorithm is given in table 3.2.

3.5 Results

Four different datasets were tested in this section, three consisting of real world data for visual evaluation of the results and another consisting of synthetic data where ground truth is available.

Since scale orthographic cameras are used, the results obtained are only known up to a translation, rotation and scale factor, thus this must be quotiented out of the numeric comparison function. To that effect all numeric error measures of the reconstructed points $\mathcal{Q}$ are computed by solving the

Coordinate Descent for Improving Surface Unfolding Solution

Input: Observations $\mathcal{P}^k$ and initial estimates for $\left\{s^k\right\}_1^K$ and $\mathcal{Q}$.
Output: Improves the value of the cost function in equation (3.9) by providing better candidates for $\left\{s^k\right\}_1^K$, $\mathcal{Q}$, and $\{\mathbf{O}_i^k : \forall i, k\}$.

1: Solve equation (3.9) only for the sub-Stiefel matrices. This is a sub-Stiefel Procrustes problem that can be solved as described in appendix C.
2: Solve for $\mathcal{Q}$. This is a least squares problem.
3: Solve for $\left\{s^k\right\}_1^K$. This is a least squares problem. If any s^k is negative, change the sign so it becomes positive and commute the sign of $\left\{\mathbf{O}_i^k : \forall i\right\}$.
4: Repeat all until some stopping condition is met (e.g. the cost function improvement is less than a certain amount).

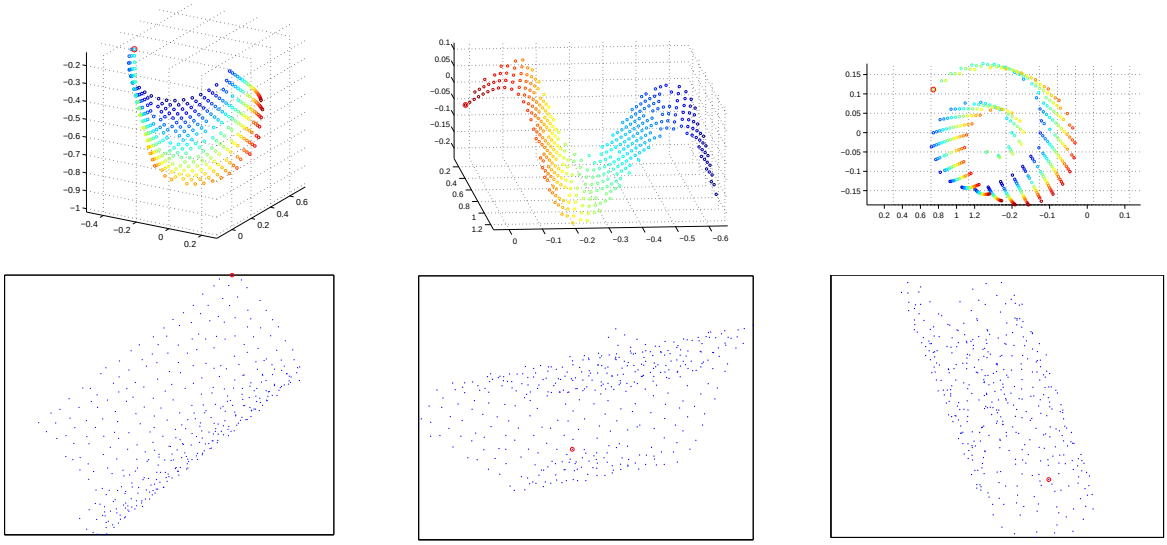
Table 3.2: Coordinate Descent for Improving Surface Unfolding Solution.

optimization problem

$$E_{\hat{\mathcal{Q}}}(\mathcal{Q}) = \begin{array}{rl} \text{minimize} & \frac{1}{N}\left\|s\mathbf{R}\mathbf{Q}(\mathbf{I}_N - \mathbf{1}_N\mathbf{1}_N^T/N) - \hat{\mathbf{Q}}\right\|^2 \\ \text{s.t.} & \mathbf{R} \in \mathsf{SO}(3) \\ & s \in \mathbb{R} \end{array}$$

where $\hat{\mathcal{Q}}$ is a zero mean and unit variance point cloud that represents ground truth. The matrices $\mathbf{Q}$ and $\hat{\mathbf{Q}}$ are any ordering of the point clouds $\mathcal{Q}$ and $\hat{\mathcal{Q}}$ where each point is represented by a column. The optimization variable $\mathbf{R}$ removes the rotational ambiguity while the scale factor s, together with the unit variance of the ground truth information, remove the scale ambiguity. Translation invariance is guaranteed by the zero mean property of the point cloud $\hat{\mathcal{Q}}$ and the mean removal projector $\mathbf{I}_N - \mathbf{1}_N\mathbf{1}_N^T/N$ applied to the points to be compared. Here $\mathbf{I}_N$ is the $N \times N$ identity matrix and $\mathbf{1}_N$ is the N dimensional column vector filled with ones.

The first dataset consists of a grid of 20×20 points wrapped in random configuration around shapes such as a cylinder, a sinusoidal surface and, as is traditionally known, a swiss roll (see figure 3.2 for a sample). The original grid is assumed to ocuppy one square unit where each side of the grid is one unit. The resulting 3D clouds of points are orthographically projected on a random plane, Gaussian noise with a given variance is added to these projections and finally the whole image is scaled by a random scale factor, obeying the scale orthographic camera model. The images are generated before each experiment, meaning that each has different orientations and bend directions as well as noise and missing points (where applicable).



(a) half cylinder (b) sine wave (c) swiss roll

Figure 3.2: Input data wrapped around different shapes. Each synthetic set contains a varying number of images similar to the ones shown. On top a 3D image is shown which is then projected on a plane, Gaussian noise is added and a scale factor is applied to generate the images on the bottom.

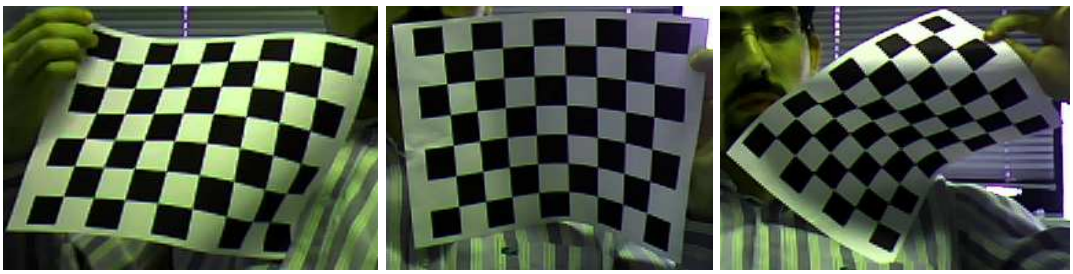


Figure 3.3: Real world data acquired with a low-resolution webcam. The dataset consists of 7 images of which 3 are shown.

The second and third datasets consist of images taken of checkered paper. In the first case, 7 images were obtained using a laptop webcam at different distances. The corner features were then identified and fed to the algorithm. This dataset does not contain occlusions of the points. Figure 3.3 shows 3 of the images used. In figure 3.4 another 3 images are shown of a different dataset consisting of 17 images of a higher sampled grid at a higher resolution.

A third real world dataset consists of 12 images of a bed cover taken at various angles and differently folded as seen in figure 3.5. Here, 118 different points were hand clicked in each image (when visible) and the algorithm was run on them. This data set provides real world data with a non-constant distribution where in some areas the sampling is not very dense for the amount of bending. There is also no hard guarantee that the embeddings truly obey the isometric properties since cloth is easily

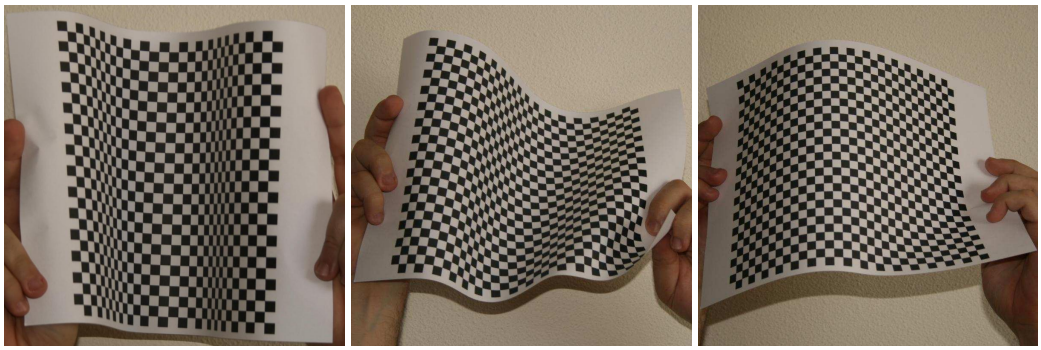


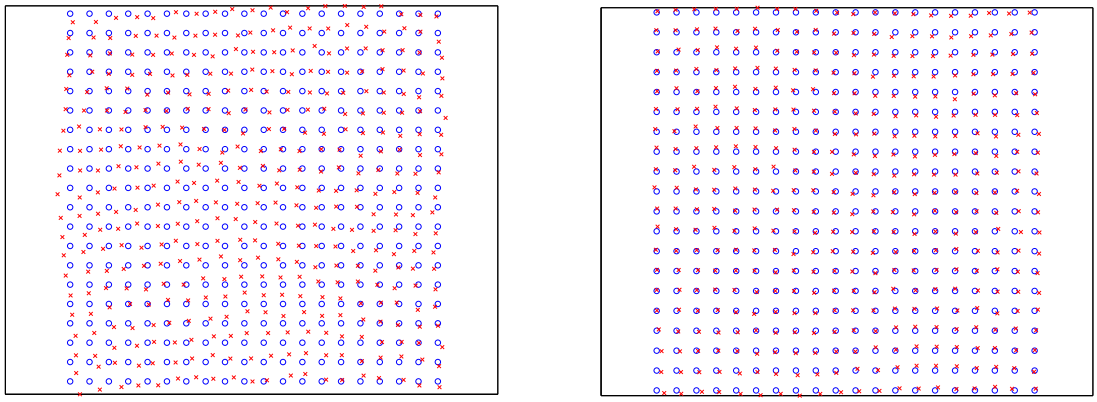
Figure 3.4: Real world data acquired with a consumer 10 megapixel camera. The dataset consists of 17 images of which 3 are shown.

Figure 3.5: A bedcover made of cloth acquired with a consumer 5 mega pixel camera. The dataset consists of 12 images, of which 3 are shown.

sheared.

Although the synthetic data set is used primarily to provide quantitative results, this section starts by showing the algorithm in action. First a set of 18 images as in figure 3.2 were generated, with a significant amount of Gaussian noise added (standard deviation equal to half the inter-grid distance) but no missing data. These images were then fed to the algorithm in two batches: one where only 6 images were used, the other using all 18 images. Figure 3.6 plots the results for visual interpretation. As expected, an increase in the number of images helps reduce the amount of noise in the reconstruction.

The visual validation is useful to provide confidence in the results, but the synthetic data is much more useful to provide quantitative expected results. With this in mind, several experiments were run on the synthetic dataset with different parameters: noise variance, number of images and percentage of missing data. Each experiment was run 10 times and the median of the result was taken when plotting the results shown in figure 3.7. In the figures it is clear that there are two sources of errors: the Gaussian noise of each feature point and the non-planar approximation. For noise levels $\sigma_{noise} = 0$ and $\sigma_{noise} = 0.001$ (remember that each side of the unfolded surface is considered to be 1 unit) there is no difference in performance, a hint that the non-planar approximation is limiting the performance of the algorithm. Only when noise levels become of the order $\sigma_{noise} = 0.01$ do the results start to



(a) 6 Images. (a) 18 Images.

Figure 3.6: Result of running the algorithm on the synthetic dataset with 6 and 18 images (red crosses). Ground truth is provided in blue circles for comparison.

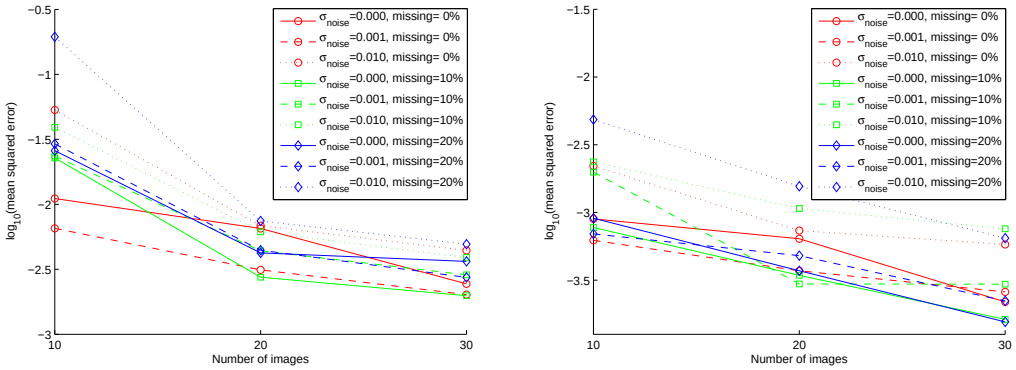


Figure 3.7: Results of applying the algorithm with different parameters. On the left is shown the result before applying the coordinate cycling algorithm (just the local algorithm), and on the right is shown the results after applying the coordinate cycling algorithm.

deteriorate due to noise, hence becoming the major limiter of performance. In either case, adding more images is a way of improving the results. Also notice that applying the coordinate cycling algorithm improves the solution by about a factor of 10.

Note that the previous discussion refers to datasets where new images are used before each experiment. This means that the projection plane used in each image, generating more or less degenerate data, and the position of the missing data points is chosen independently in each experiment. Another useful comparison is when a single dataset is generated and several parameters are changed in this dataset. Figure 3.8 shows this case. A single dataset of 10 images is generated and before each experiment noise is added and random points are removed. As the results show, Gaussian noise does not affect performance as much as missing data. The results also show that 10% missing data (unless pathological cases are considered where the data degenerates) does not impact performance

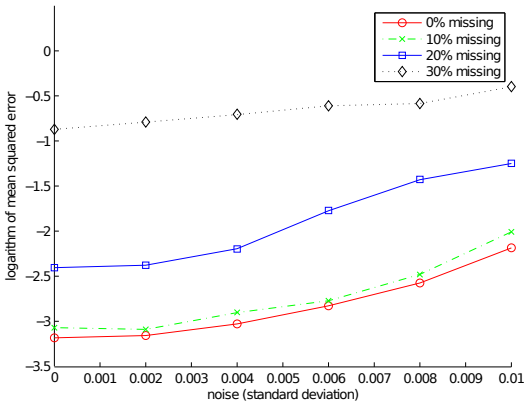
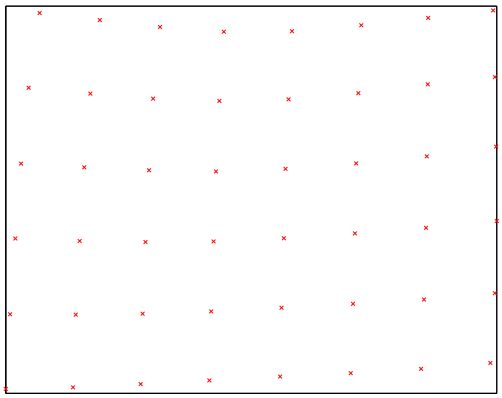


Figure 3.8: Logarithmic mean squared error as a function of Gaussian noise with a given standard deviation and missing data percentage. Each test is performed with 10 synthetic images and 10 neighbors. The results shown represent the median of 5 experiments, where noise and missing data are chosen randomly before each experiment.

(a) Sample acquired image.



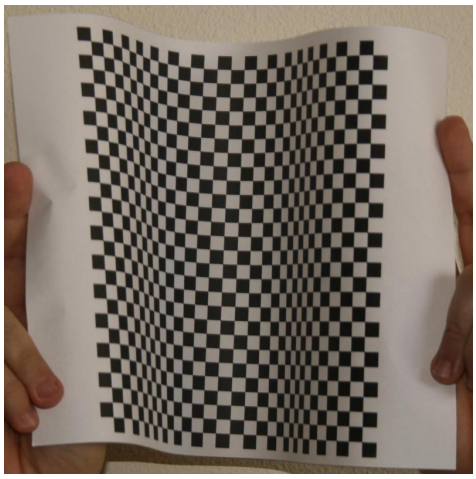
(b) Results obtained.

Figure 3.9: Results of applying the algorithm to 7 camera acquired images.

significantly, but after 20% performance begins to drop rapidly.

The results obtained when applying the algorithm to the first real world example are shown in figure 3.9. Note that globally the reconstruction appears to have a slight pinch in the middle. This is caused by the features not being dense enough for the amount of distortion introduced and the number of images being too low. When the sampling density is increased, as well as the number of images, better results are obtained as shown in figure 3.10.

The final dataset, consisting of images of a bed cover, provides a more realistic real world example. The qualitative results are shown in figure 3.11. The results are mostly correct, except for a slight skew that exists in "almost isolated islands" such as the one on the right middle consisting of 5 points. These features are neighbors of each other and are linked to the rest of the features by only a few neighboring



(a) Sample acquired image.

(b) Results obtained.

Figure 3.10: Results of applying the algorithm to 17 camera acquired images.

connections. This allows for offsets to occur, even though locally the reconstruction is correct.

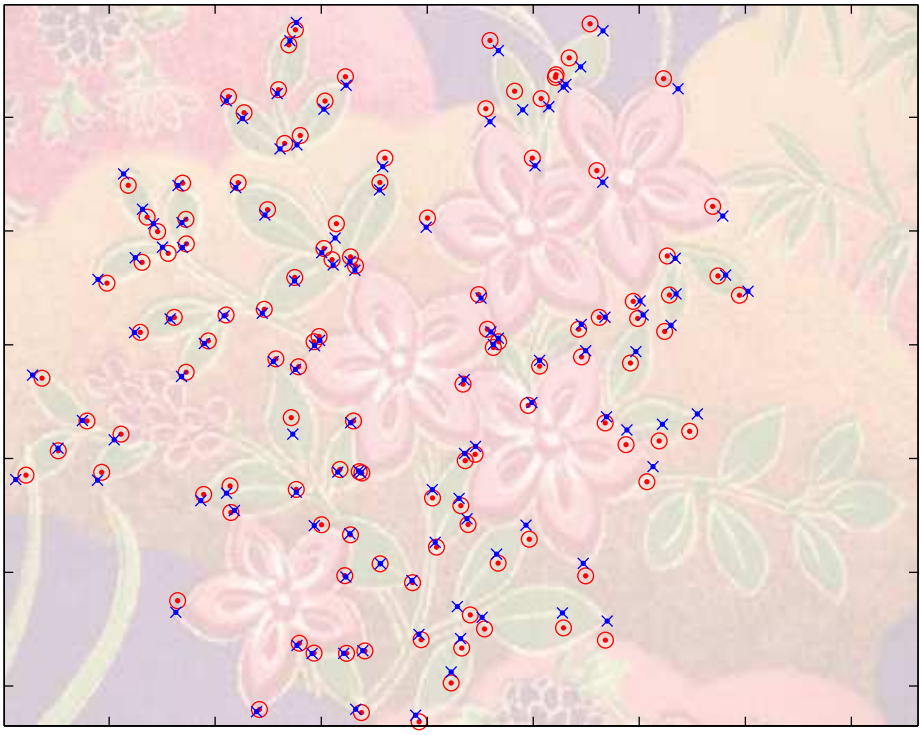


Figure 3.11: Reconstruction of the bed cover cloth overlaid on an image taken of the cloth laying flat. Blue crosses are the reference clicked points, red circles are the results given by the algorithm, both are overlaid on a picture taken of the flatbed cover (not in the dataset).

Chapter 4

The Pose Estimation Problem

Contents

The **pose estimation** problem proposes to recover the actual 3D points $\mathbf{r}_i^k \in \mathcal{R}^k \subset \mathcal{S}^k$. It is assumed that besides the observations $\mathbf{p}_i^k$ the surface points $\mathbf{q}_i$ are also available. The proper problem statement is

Problem Statement 4.1 (Pose Estimation Problem) *Starting with sets $\mathcal{P}^k \subset \mathbb{R}^2$ of image point observations and $\mathcal{Q} \subset \mathbb{R}^2$ known to have been generated by the model*

$$\mathbf{p}_i^k = C^k\left(\mathcal{I}^k\left(\mathbf{q}_i\right)\right) = \begin{cases} \mathbf{r}_i^k = \mathcal{I}^k\left(\mathbf{q}_i\right) \\ \mathbf{p}_i^k = C^k\left(\mathbf{r}_i^k\right) \end{cases} \qquad \forall i,k \tag{4.1}$$

recover the 3D embedded points $\mathbf{r}_i^k \in \mathcal{R}^k \subset \mathcal{S}^k \subset \mathbb{R}^3$ under the following conditions:

- *C^k represents the projection model of a scale-orthographic camera (see assumption 2.2).*
- *$\mathcal{I}^k : \mathbb{R}^2 \to \mathbb{R}^3$ are isometric embedding functions.*
- *All points have been previously matched and the correspondence between points in different $\mathcal{P}^k$ and $\mathcal{Q}$ is known (see assumption 2.1 and the discussion immediately after).*
- *$\mathbf{r}_i^k \in \mathcal{R}^k$ are represented in the corresponding camera frame.*

Since the $\mathbf{q}_i$ are now known, the problem decouples and each image can be treated separately. The last condition in the problem statement specifies the coordinate frame where the points $\mathbf{r}_i^k \in \mathcal{R}^k$ in

equation (4.1) are represented. Since scale orthographic cameras are assumed, the camera model is thus

$$\mathbf{p}_i^k = s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{r}_i^k. \tag{4.2}$$

On the other hand, writing equation (4.1) locally in terms of vectors as suggested in chapter 2 yields that

$$\mathbf{u}_i^k = \mathbf{J}_i^k \mathbf{w}_i. \tag{4.3}$$

where $\mathbf{u}_i^k \in \mathbb{T}_{\mathbf{r}_i^k}\mathcal{R}^k$ and $\mathbf{w}_i \in \mathbb{T}_{\mathbf{q}_i}\mathbb{R}^2$. Fact 2.3 guarantees that $\mathbf{J}^k$ is Stiefel. Remembering that in chapter 3 the sub-Stiefel matrices $\mathbf{O}_i^k$ were defined as the multiplication of the camera projection matrix and this Jacobian, results in this case that they are in fact the top-most 2×2 entries of matrix $\mathbf{J}_i^k$:

$$\mathbf{J}_i^k = \begin{bmatrix} \mathbf{O}_i^k \\ * \, * \end{bmatrix},$$

where the stars $*$ are placeholders for unknown entries.

If this problem is considered after solving the surface unfolding problem as described in chapter 3 matrices $\mathbf{O}_i^k$ and the scalars s^k are known. If not, a way of obtaining these solely from $\mathbf{p}_i^k$ and $\mathbf{q}_i$ will be presented in section 4.1. In either case, consider that these values are known.

From the Stiefel set condition $\left(\mathbf{J}_i^k\right)^T \mathbf{J}_i^k = \mathbf{I}_2$ the missing entries can be found up to a sign ambiguity:

$$\mathbf{J}_i^k = \begin{bmatrix} \mathbf{O}_i^k \\ \pm \mathbf{o}_i^k \end{bmatrix}. \tag{4.4}$$

these entries, represented as $\mathbf{o}_i^k$ can be chosen uniquely if one admits that the first non-zero entry must be positive, but what follows does not depend on what choice is made.

Going back to equation (4.3), and once again approximating the vectors $\mathbf{u}_i^k$ by the differences of points

$$\mathbf{r}_{j_i}^k - \mathbf{r}_i^k \approx \mathbf{J}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i\right), \tag{4.5}$$

a new equation is obtained which, together with equation (4.2), defines a system of equations that should be satisfied. Unfortunately, the Jacobian matrices $\mathbf{J}_i^k$ are not completely known and an integral constraint appears (the unknown sign in equation (4.4)).

If the coordinates of $\mathbf{r}_i^k$ are written explicitly as

$$\mathbf{r}_i^k = \begin{bmatrix} x_i^k \\ y_i^k \\ z_i^k \end{bmatrix}$$

the complete system of equations becomes

$$s^k \begin{bmatrix} x_i^k \\ y_i^k \end{bmatrix} = \mathbf{p}_i^k \quad \forall (i,j,k) \in \mathcal{V} \tag{4.6}$$

$$\begin{bmatrix} x_{j_i}^k \\ y_{j_i}^k \end{bmatrix} - \begin{bmatrix} x_i^k \\ y_i^k \end{bmatrix} \approx \mathbf{O}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i \right) \quad \forall (i,j,k) \in \mathcal{V} \tag{4.7}$$

$$z_{j_i}^k - z_i^k \approx a_i^k \mathbf{o}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i \right) \quad \forall (i,j,k) \in \mathcal{V} \tag{4.8}$$

$$(a_i^k)^2 = 1 \quad \forall i,k \tag{4.9}$$

where a_i^k are the unknown sign of $\mathbf{o}_i^k$.

The first thing to notice when solving this system of equations is that the first two equations with variables x_i^k and y_i^k are completely independent of the last two equations in variables z_i^k and a_i^k, hence they can be solved separately. The first system is an over constrained system of linear equations which can be solved in closed form as a least squares problem. Since the topmost equation depends only on the image points, while the second depends only on the template points, it's easy to include prior information when solving the system. For example if the template is known to be error free and the sampling is dense, it makes sense to give the second equation more importance than the first. These probabilistic approaches are beyond the scope of this thesis.

Unfortunately, the second system is not so simple due to the quadratic equations. The strategy taken to get an approximate solution is to relax the system as:

$$z_{j_i}^k - z_i^k \approx a_i^k \mathbf{o}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i \right) \quad \forall (i,j,k) \in \mathcal{V} \tag{4.10}$$

$$\prod_i (a_i^k)^2 = 1 \tag{4.11}$$

It might not be immediately clear why this system is easier to solve than the original one, until one realizes that the linear equation's solution is given up to scale. Taking this into account, one can first find a solution that satisfies the linear equation, and then scale the result to satisfy the second equation.

Collecting all the variables of equation (4.10) into a vector

$$\mathbf{x}^k = \begin{bmatrix} z_1^k & z_2^k & \dots & z_n^k & a_1^k & a_2^k & \dots & a_n^k \end{bmatrix}^T,$$

the equation is linear, meaning that it can be written as

$$\left(\mathbf{l}_{ij_i}^k\right)^T \mathbf{x}^k \approx \mathbf{0} \quad \forall (i,j,k) \in \mathcal{V}, \tag{4.12}$$

where $\mathbf{l}_{ij_i}^k$ is a constant vector, which describes the linear equation.

Collecting the constraints to all visible pairs of points in image k into a matrix, the following system is obtained

$$\mathbf{L}^k \mathbf{x}^k \approx \mathbf{0}, \tag{4.13}$$

This system has a zero singular value associated with the singular vector collinear with $\begin{bmatrix} \mathbf{1}_{1\times n} & \mathbf{0}_{1\times n} \end{bmatrix}$, which is a non-interesting trivial solution due to the fact that an orthographic camera is insensitive to depth. The second singular value whose corresponding singular vector is the solution up to a scale factor.

The solution vector should have the entries corresponding to the $\hat{a}_i^k$ all of equal magnitude, but in the presence of noise this will not occur and the variance of these will provide a hint as to the quality of the data. From an optimization perspective this is the solution to the singular value problem

$$\begin{aligned} \text{minimize} \quad & (\mathbf{x}^k)^T (\mathbf{L}^k)^T \mathbf{L}^k \mathbf{x}^k \\ \text{s.t.} \quad & (\mathbf{x}^k)^T \mathbf{x}^k = 1 \\ & \begin{bmatrix} \mathbf{1}_{1\times n} & \mathbf{0}_{1\times n} \end{bmatrix} \mathbf{x}^k = 0 \end{aligned}$$

Since the solution was given up to a scale factor, equation (4.11) can now be used to fix the scale. Applying the logarithm to both sides of the equation allows for the scale factor c^k to be found:

$$\prod_i \left((c^k)^2 (\hat{a}_i^k)^2 \right) = 1 \iff N \log\left(\left|c^k\right|\right) + \sum_i \log(\left|\hat{a}_i^k\right|) = 0 \tag{4.14}$$

$$\iff \log(\left|c^k\right|) = -\sum_i \log\left(\left|\hat{a}_i^k\right|\right) / N \tag{4.15}$$

and c^k is used to scale the whole solution vector $\mathbf{x}^k$. The fact that orthographic cameras cannot distinguish depth is present here once again in the fact that c^k can only be computed up to a sign. Unfortunately, in the presence of noise this process can be very ill-conditioned. Instead, the previous

step will only be used to infer the sign of a_i^k, which will be forced to be either 1 or -1. After these have been found, equation (4.10) is solved again where only the $z_k^i \in \mathbb{R}$ are unknowns, resulting in a linear system of equations.

4.1 Cold Start

The method shown hinges on knowing s^k and $\mathbf{O}_i^k$ in advance. If this method is to be run after the solution for the surface unfolding problem is obtained as described in chapter 3, these values are already available. If not, they need to be pre-computed from the knowledge of sets $\mathcal{Q}$ and $\mathcal{P}^k$ only. To do so, the already discussed equation 3.3, here repeated for convenience,

$$\mathbf{p}_{j_i}^k - \mathbf{p}_i^k \approx s^k \mathbf{O}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i\right) \quad \forall (i,j,k) \in \mathcal{V}, \tag{4.16}$$

fits into this discussion perfectly. Collecting all the neighboring points together, the previous may be written as

$$\mathbf{V}_i = \begin{bmatrix} \mathbf{p}_{1_i}^1 - \mathbf{p}_i^1 & \mathbf{p}_{2_i}^1 - \mathbf{p}_i^1 & \dots & \mathbf{p}_{N_{ii}}^1 - \mathbf{p}_i^1 \end{bmatrix} \qquad \mathbf{M}_i^k = \begin{bmatrix} s^k \mathbf{O}_i^k \end{bmatrix}$$
$$\mathbf{W}_i = \begin{bmatrix} \mathbf{q}_{1_i} - \mathbf{q}_i & \mathbf{q}_{2_i} - \mathbf{q}_i & \dots & \mathbf{q}_{N_{ii}} - \mathbf{q}_i \end{bmatrix}$$

The easiest way to solve this system is to first relax the matrices $\mathbf{M}_i^k$ to be in $\mathsf{M}^{2\times 2}$ and solving the least squares problems to obtain $\hat{\mathbf{M}}_i^k$ and obtain estimates for s^k using the relaxations

$$\prod_i \left(\sigma_{\max}(\hat{\mathbf{M}}_i^k)/s^k \right) = 1$$

which is solved by applying logarithms to both sides of the equation. Once initial estimates for s^k are available, the optimization problem

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i,j,k)\in\mathcal{V}} \left\| \left(\mathbf{p}_{j_i}^k - \mathbf{p}_i^k \right) - s^k \mathbf{O}_i^k \left(\mathbf{q}_{j_i} - \mathbf{q}_i \right) \right\|_2^2 \\ \text{s.t.} \quad & s^k \in \mathbb{R}^+ \\ & \mathbf{O}_i^k \in \mathbb{SS} \end{aligned}$$

is solved by coordinate cycling (or a gradient descent method) in a similar way to what was done in section 3.4. Here this problem is simplified since $\mathbf{q}_i$ are already known. Solving for s^k is a simple least squares problem, while solving for $\mathbf{O}_i^k$ is the sub-Stiefel Procrustes problem described in appendix

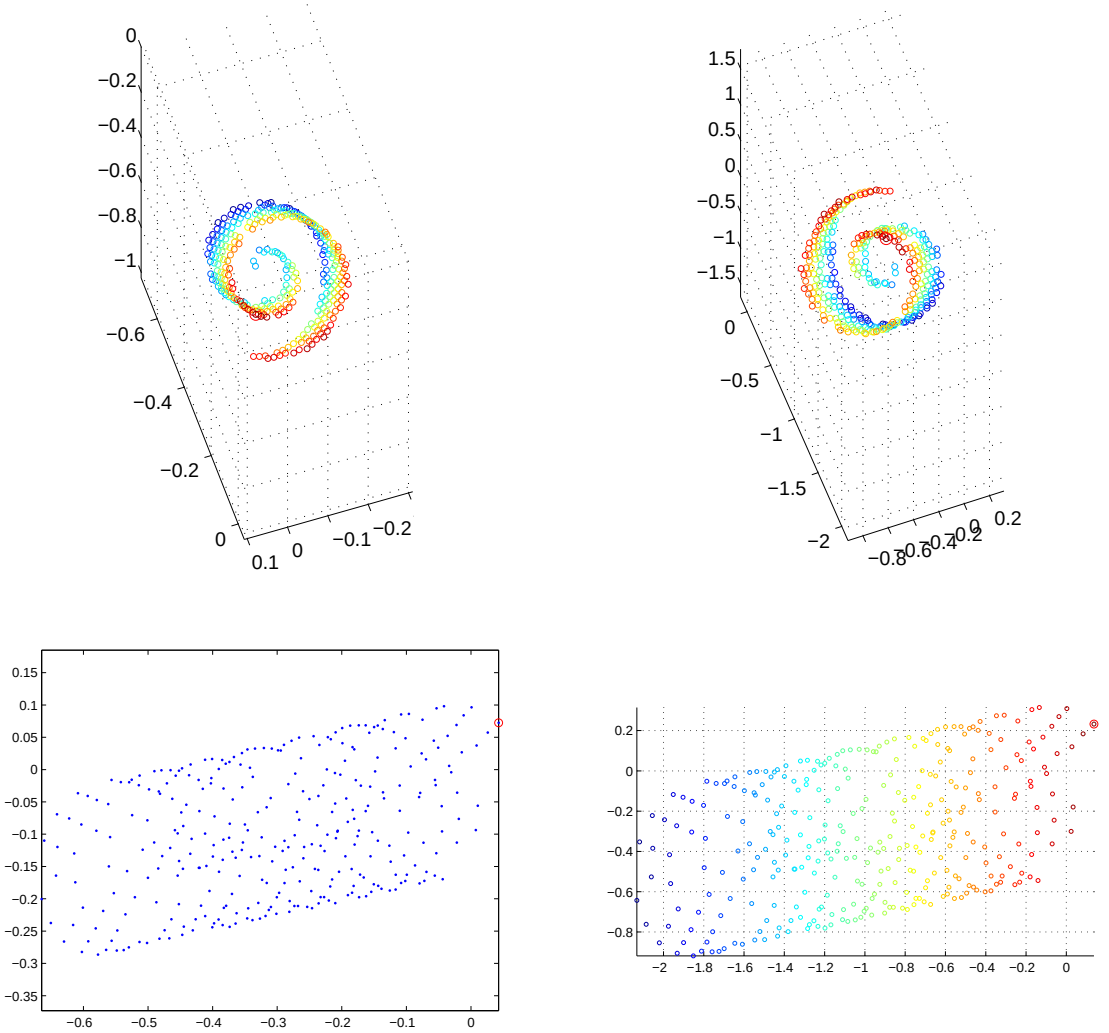
C. If during the coordinate cycling any of the s^k becomes negative, the sign can be absorbed by the sub-Stiefel matrices.

4.2 Results

The results shown here use the same synthetic dataset as described in section 3.5. The results obtained can be interpreted with mixed conclusions. The main problem stems from the fact that the signs a_i^k cannot be uniquely estimated with a scale orthographic camera model. Figure 4.1 shows a reconstruction where all the signs (up to a global sign change) were correctly obtained. The reconstructed 3D cloud is visually similar to the 3D cloud that generated the image used. Unfortunately, in the same dataset (exactly the same conditions), some of the point clouds were not correctly reconstructed as shown in figure 4.2. Note that in this example, the retro-projected image of the reconstructed point cloud is correct (compare the two bottom images), so the problem is due to the solution not being unique.

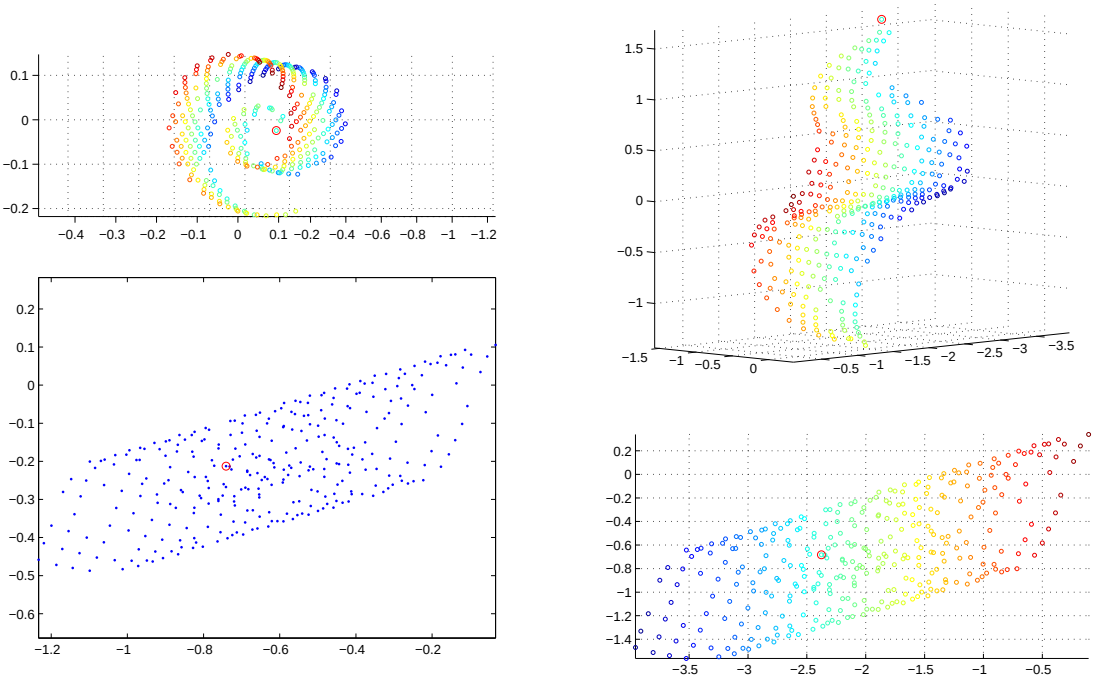
In the case of reconstructions with a high noise level the same problem occurs but the results can be catastrophic. Figure 4.3 presents some results where the algorithm was able to recover the pose, while figure 4.4 presents some of the bad reconstructions.

Figures 4.5 and 4.6 show the results of applying this algorithm to high resolution real world images described in section 3.5. The quality of the reconstruction should not go unnoticed when compared with the previous synthetic images and it is due to the fact that real world noise, under perfect conditions, is nowhere as large as the noise applied to the synthetic examples.



(a) Original 3D point cloud and projected images. (b) Two views of the same reconstructed cloud.

Figure 4.1: Applying the pose estimation algorithm to an image of a set of 30 where there are 20% missing data and the noise standard deviation is 0.001 (where the inter-grid distance is 1 unit). The bottom right image is seen from the same angle as the input image shown on the bottom left.



(a) Original 3D point cloud and projected images. (b) Two views of the same reconstructed cloud.

Figure 4.2: Applying the pose estimation algorithm to an image of a set of 30 where there are 20% missing data and the noise standard deviation is 0.001 (where the inter-grid distance is 1 unit). The bottom right image is seen from the same angle as the input image shown on the bottom left.

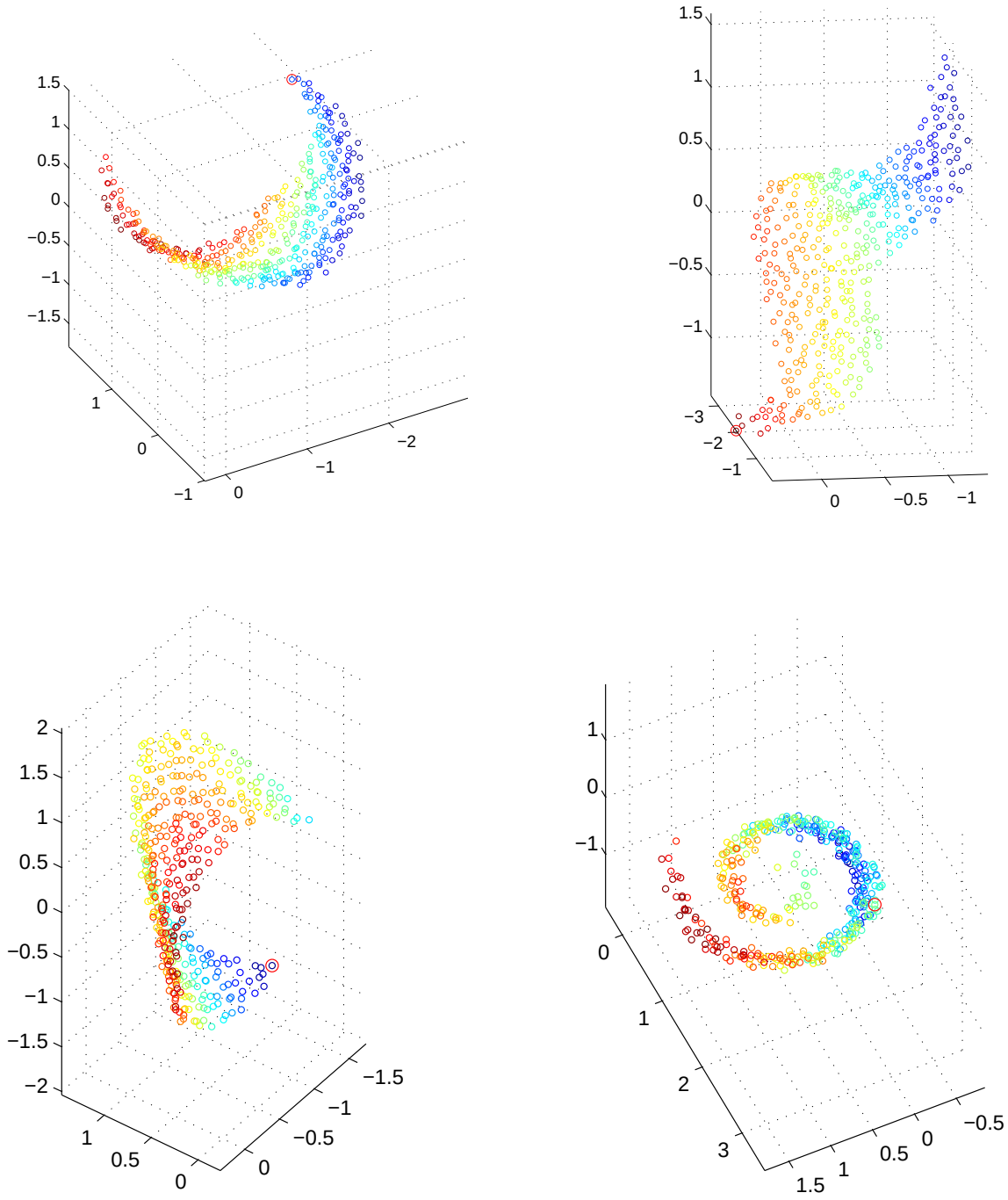


Figure 4.3: Four different reconstructions of a dataset with 30 images and a high noise level ($\sigma_{noise} = 0.01$). On the left two half cylinder reconstructions, on the top right a sine wave and on the bottom right a swiss roll.

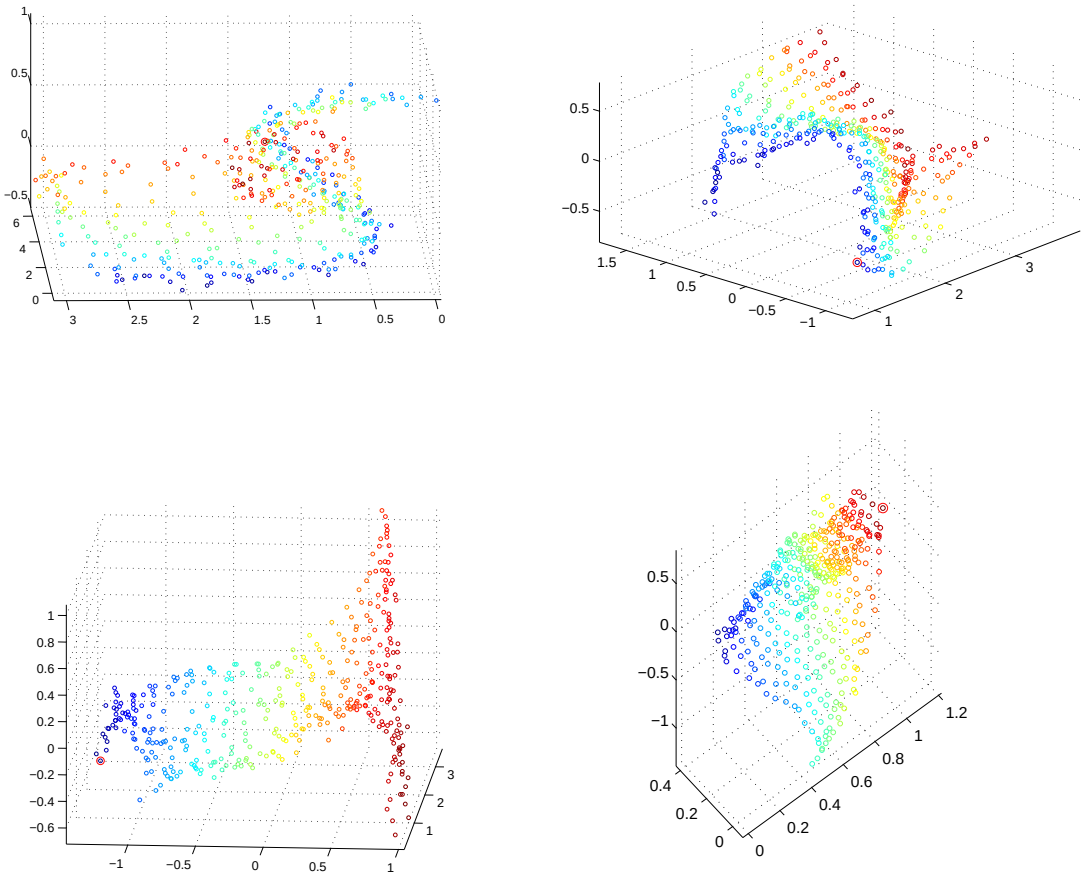


Figure 4.4: Four different reconstructions of a dataset with 30 images and a high noise level ($\sigma_{noise} = 0.01$) when the reconstruction fails completely. Top 2 images: sine wave; Bottom left: half cylinder; Bottom right: swiss roll.

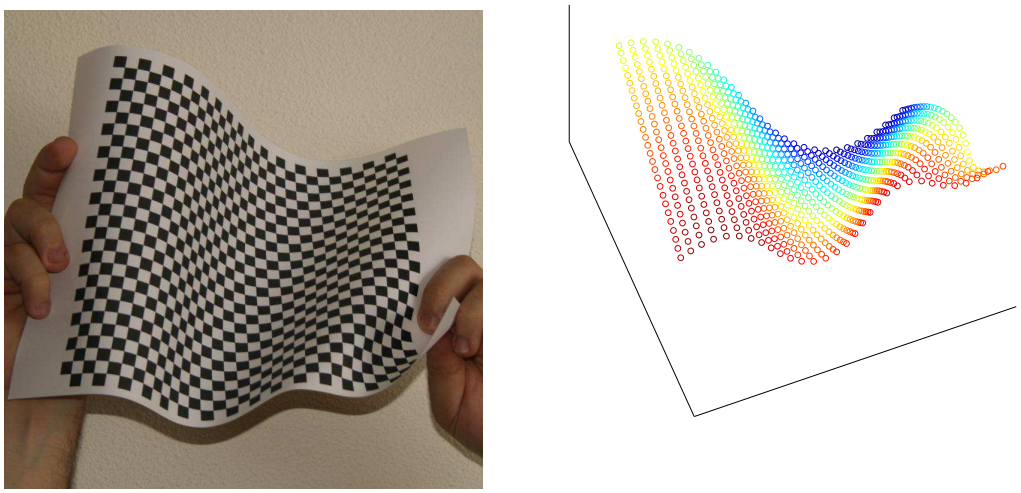


Figure 4.5: Original image and reconstructed 3D point cloud.

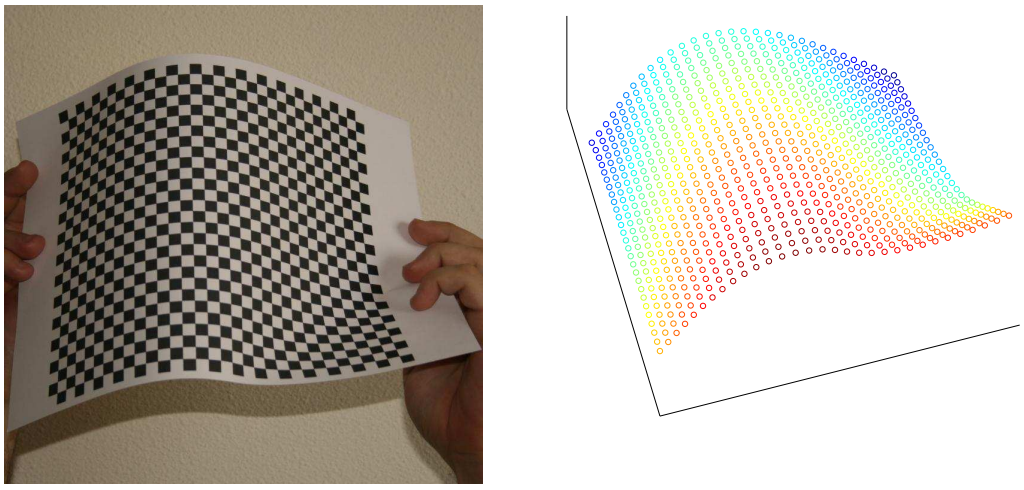


Figure 4.6: Original image and reconstructed 3D point cloud.

Chapter 5

Isometry Estimation Problem

Contents

The last two chapters have built themselves around a local property of the isometric embedding functions $\mathcal{I}$ (using fact 2.3). This property has allowed for surface unfolding and pose estimation to be performed with decent results. This chapter intends to go a step further by asking and fully answering the questions “can further properties of isometry functions be used and can these functions be somehow parameterized?”, providing a constructive way of building developable surfaces as the image of isometry functions.

5.1 Torsal Ruled Surfaces

A **Ruled surface** is defined as the image of a set $\mathcal{C} \subset \mathbb{R}^2$ (assume it's convex to simplify the discussion) through a function

$$r : \mathcal{C} \to \mathbb{R}^3 \quad r(t, v) = c(t) + vd(t)$$

where $c : \mathcal{T} \subset \mathbb{R} \to \mathbb{R}^3$ and $d : \mathcal{T} \to \mathbb{R}^3 - \{\mathbf{0}\}$ and $\mathcal{T}$ is the projection of $\mathcal{C}$ on the first coordinate. The line (segments) defined as the image of the function

$$l_t(v) = c(t) + vd(t)$$

are called the **rulings** of the surface.

When talking about surfaces in $\mathbb{R}^3$ one usually refers to embedded surfaces. Unfortunately guaranteeing injectivity is not trivial and so in this section only an immersion is guaranteed which is a local property. In the previous context it is easy to impose by stating that $\dot{c} + v\dot{d}$ and d must be linearly independent in the image of $\mathcal{C}$. Although not stated previously it is implied in the previous sentence that the functions c and d have to be smooth. Without loss of generality, $d(t)$ can be forced to be unit normed (with an appropriate change of the set $\mathcal{C}$), at which point it is called the **directrix** of the surface.

The parameterization defines a surface which is locally isometric to a plane when $\dot{c}$, d and $\dot{d}$ are everywhere linearly dependent (see the discussion after preposition 4 in chapter 5 of [39] or equation 9 in section 3.5 of [7]). In this case it is called a **torsal ruled surface**.

Fact 5.1 (Torsal Ruled Surface) *Denote the union of $\mathbb{R}$ with plus and minus infinity as $\bar{\mathbb{R}}$. Starting from a set $\mathcal{T} \subset \mathbb{R}$ and two functions $\alpha : \mathcal{T} \to \bar{\mathbb{R}}$ and $\beta : \mathcal{T} \to \bar{\mathbb{R}}$ such that $\alpha(t) < \beta(t)$, define a set $\mathcal{C} = \{(t, v) : t \in \mathcal{T}, v \in]\alpha(t), \beta(t)[\}$.*

*A **torsal ruled surface** is defined as the image the set $\mathcal{C}$ through the function*

$$r : \mathcal{C} \to \mathbb{R}^3 : r(t, v) = c(t) + vd(t)$$

where $c : \mathcal{T} \subset \mathbb{R} \to \mathbb{R}^3$ and $d : \mathcal{T} \subset \mathbb{R} \to \mathbb{S}(2) \subset \mathbb{R}^3$ such that for every $(t, v) \in \mathcal{C}$

- *[immersion condition] $\dot{c} + v\dot{d}$ and d are linearly independent.*
- *[torsal ruled surface condition] $\dot{c}$, d and $\dot{d}$ are linearly dependent.*

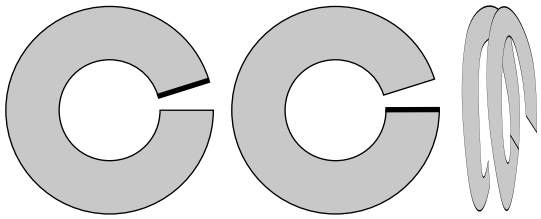


Figure 5.1: A developable surface which is not a global isometry of a planar subset. If the two cut disks are glued together along the bold edge, the resulting surface (on the right) cannot be flattened onto a plane without overlap.

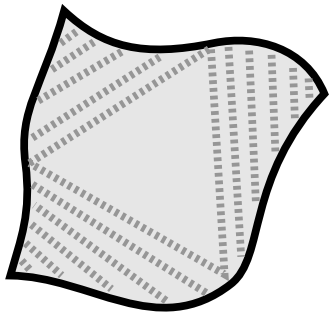


Figure 5.2: This plane isometry is a triangular sheet of paper whose vertices have been smoothly bent. Each of the 3 bent parts must necessarily be torsal ruled surfaces (the rulings have been drawn), but the whole surface is not torsal ruled.

5.2 The Relation Between Isometries of planes, Torsal Ruled Surfaces and Developable Surfaces

Developable surfaces is the classic name given to surfaces which have everywhere zero Gaussian curvature. Another way of stating this definition is saying that they are locally the image of an open subset of a plane by an isometry. The set of developable surfaces is larger than the set of **plane isometries** since the first is a local property while the later is global. A classical example of a developable surface which is not an isometry of the plane is a cylinder. The cylinder has everywhere zero curvature, but it is not the image of an open subset of a plane by an isometry (it fails at the topological level). If a single straight line is removed from the cylinder then it becomes an isometry of a plane. Another way in which a developable surface might not be an isometry of a plane is if the embedded surface is large enough that the unfolding process forces overlapping (see the example in figure 5.1).

Another subclass of developable surfaces are the already mentioned **torsal ruled surfaces** which are not equivalent to planar isometries. The example shown in figure 5.1 is a torsal ruled surface which is not a planar isometry, and figure 5.2 illustrates a planar isometry which is not torsal.

Figure 5.3 represents the mentioned classes graphically. This text is interested in plane isometries

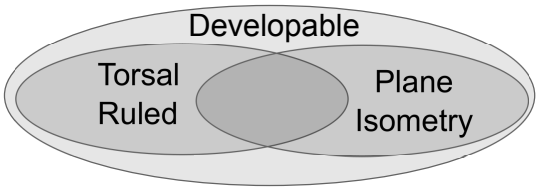


Figure 5.3: The three classes of zero curvature surfaces mentioned in the text.

but unfortunately no direct way of using them is known. On the other hand, torsal ruled surfaces are easy to work with, but the overlap with plane isometries is not perfect. Looking at the counter example in figure 5.2 one notices that although the whole surface is not torsal ruled, it is the union of 3 torsal ruled surfaces and a planar set (the 3 corners plus the planar triangle at the center). This loose statement constitutes the basis of what's known as the "classification" of developable surfaces. See chapter 5 of [39] for a detailed exposition. The main result is that smooth developable surfaces can be generated as unions, along a common straight line segment, of torsal ruled surfaces and planar sets. Although planar sets can be parameterized as a torsal ruled surface, since the rulings are not canonically defined on the surface they are treated separately.

The rest of this chapter considers that the plane isometry is also a torsal ruled surface that contains no planar sections on the embedded surface. To deal with general surfaces is considered to be future work.

5.2.1 Redundancy of the Torsal Ruled Surface Description

The description of torsal ruled surfaces provided in fact 5.1 is far from unique. In fact, two different means of changing the parameters $(\mathcal{T}, \alpha, \beta, c, d)$ which define the surface can be readily identified by re-parameterizing the variables (t, v).

If the curve c is changed to any other curve that intersects the same ruling at each parameter t (see figure 5.4), a new parameterization for the surface is obtained. Given a function $f : \mathcal{C} \to \mathbb{R}$, this can be expressed as

$$\hat{r}(t, v) = c(t) + (v + f(t))d(t) = \underbrace{c(t) + f(t)d(t)}_{\hat{c}(t)} + vd(t).$$

The under-brace makes it clear that a new parameterization for the same surface is obtained, as long as the functions α and β are changed as $\hat{\alpha} = \alpha - f$ and $\hat{\beta} = \beta - f$, thus inducing a new set $\hat{\mathcal{C}}$. Note that the directrix d and the set $\mathcal{T}$ are left unchanged. One still has to make sure that the immersion and the torsal ruled surface properties hold for the new parameterization.

To check the immersion condition, one needs to make sure that $\dot{\hat{c}} + v\dot{d} = \dot{c} + \dot{f}d + f\dot{d} + v\dot{d}$ is

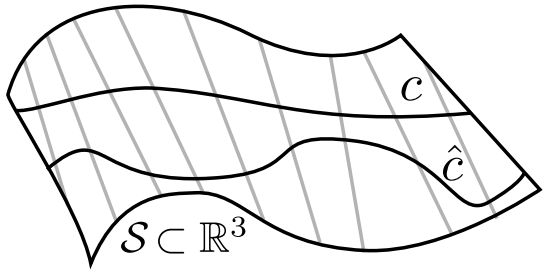


Figure 5.4: Redundancy of the curve c. The curve $\hat{c}$ is can also be used to define the same torsal ruled surface.

linearly independent of d on $\hat{\mathcal{C}}$. This is true because $\dot{c} + (f + v)\dot{d}$ is linearly independent of d on the set $\hat{\mathcal{C}}$ (this is exactly the condition on the original parameterization), and adding $\dot{f}d$ does not change the interdependent property.

To check the torsal ruled surface condition, one simply notes that if $\dot{c}, d$, and $\dot{d}$ are linearly dependent, then $\dot{c} + \dot{f}d + f\dot{d}, d$, and $\dot{d}$ are also linearly dependent since only scaling and dependent vector additions are performed.

One way to reduce the description is to ensure that one can always find a $\hat{c}$ such that $\left\langle \dot{\hat{c}}, d \right\rangle = 0$, meaning that the curve c intersects each ruling orthogonally whenever $\dot{\hat{c}} \neq 0$, and add this condition to fact 5.1. To do so, an f must be found that obeys $\left\langle \dot{c} + \dot{f}d + f\dot{d}, d \right\rangle = 0$. Using the fact that $d(t) \in \mathbb{S}(2)$, one knows that $\langle d, d \rangle = 1$ and $\left\langle \dot{d}, d \right\rangle = 0$, hence the stated condition reads simply $\dot{f} = -\langle \dot{c}, d \rangle$. So, given an initial condition, this differential equation can be integrated to produce an f that obeys the condition. The resulting curve $\hat{c}$ is not unique since any constant can be used as an initial condition to f. Here, changing this initial condition will be called "translating along the rulings".

A consequence of imposing this condition is that it allows the immersion and torsal ruled surface conditions to be relaxed. The second condition states that it is always possible to find $a, b, c \in \mathbb{R}$ not all zero, such that $a\dot{c} + bd + c\dot{d} = 0$. By taking inner product with d one finds that $b = 0$ necessarily. Thus the condition can be relaxed to $a\dot{c} + c\dot{d} = 0$, i.e. $\dot{c}$ and $\dot{d}$ linearly dependent. Since $\left\langle \dot{d}, d \right\rangle = 0$ and $\langle \dot{c}, d \rangle = 0$, the only way for the immersion condition to fail is if $\dot{c} + v\dot{d} = 0$. Thus, the immersion condition can assume this simpler form.

The other way to change the parameterization, without changing the resulting surface is to consider a re-parameterization ϕ of the parameter t: $\hat{r}(t, v) = r(\phi(t), v)$, changing $\hat{\mathcal{T}} = \phi^{-1}(\mathcal{T})$. The immersion condition states that $\dot{c}(\phi(t))\dot{\phi}(t) + v\dot{d}(\phi(t))\dot{\phi}(t)$ must be everywhere linearly independent of $d(\phi(t))$ thus forcing $\dot{\phi}(t) \neq 0$. This means that ϕ must be a strictly increasing or a strictly decreasing function. The torsal ruled surface condition is also satisfied since only a nonzero rescaling of the

vectors occurs.

The immersion condition excludes $\dot{c}$ and $\dot{d}$ from being simultaneously 0. Hence, a way of reducing the parameterization space, is to consider these two vectors to have unit joint norm, by this meaning that the 6 dimensional vector obtained by concatenating both vectors has unit norm. This way a $\dot{\phi}$ can be fixed up to a sign. If this condition is imposed, the only re-parameterizations ϕ which are still allowed will be $\phi(t) = \pm t + k$, where k is any constant.

To summarize what has been said, given a torsal ruled surface defined by 5.1, it is always possible to find a representation $(\mathcal{T}, \alpha, \beta, c, d)$ such that:

Fact 5.2 (Normalized Torsal Ruled Surface) *Denote the union of $\mathbb{R}$ with plus and minus infinity as $\bar{\mathbb{R}}$. Starting from a set $\mathcal{T} \subset \mathbb{R}$ and two functions $\alpha : \mathcal{T} \to \bar{\mathbb{R}}$ and $\beta : \mathcal{T} \to \bar{\mathbb{R}}$ such that $\alpha(t) < \beta(t)$, define a set $\mathcal{C} = \{(t, v) : t \in \mathcal{T}, v \in]\alpha(t), \beta(t)[\}$.*

*A **normalized torsal ruled surface** is defined as the image of the set $\mathcal{C}$ through the function*

$$r : \mathcal{C} \to \mathbb{R}^3 : r(t, v) = c(t) + vd(t)$$

where $c : \mathcal{T} \subset \mathbb{R} \to \mathbb{R}^3$ and $d : \mathcal{T} \subset \mathbb{R} \to \mathbb{S}(2) \subset \mathbb{R}^3$ such that for every $(t, v) \in \mathcal{C}$

- *[orthogonality condition] $\langle \dot{c}, d \rangle = 0$.*
- *[immersion condition] $\dot{c} + v\dot{d} \neq 0$.*
- *[torsal ruled surface condition] $\dot{c}$ and $\dot{d}$ are linearly dependent.*
- *[normalizing condition] $\langle \dot{c}, \dot{c} \rangle + \left\langle \dot{d}, \dot{d} \right\rangle = 1$*

As discussed, the representation is still not unique since translations along the rulings with a constant function f is still allowed. Suppose that c and d are the functions parameterizing the surface and consider a new parameterization $\hat{c} = c + fd$ and $\hat{d} = d$, implying that $\dot{\hat{c}} = \dot{c} + f\dot{d}$ and $\dot{\hat{d}} = \dot{d}$. If the original vectors were normalized, these vectors will not. A new re-parameterization ϕ must be chosen to satisfy the normalizing condition. After that, any other ϕ of the form $\phi(t) = \pm t + k$, for any constant k can be used to generate a new normalized ruled surface parameterization without changing the surface itself.

5.2.2 Differential model

Instead of considering the curves c and d as the surface generating primitives, this section attempts a more compact notation by using their derivatives as the generating functions. The torsal ruled surface condition implies that there must exist a vector function $s : \mathcal{T} \to \mathbb{S}(2)$ and functions $\gamma : \mathcal{T} \to \mathbb{R}$ and $\delta : \mathcal{T} \to \mathbb{R}$ such that $\dot{c} = \gamma s$ and $\dot{d} = \delta s$. The normality condition imposes that there exists a function $z : \mathcal{T} \to \mathbb{R}$ such that $\gamma(t) = \cos(z(t))$ and $\delta(t) = \sin(z(t))$.

The vector function s is orthogonal to d which can be seen both from the orthogonality condition and the fact that d is a curve on a sphere which implies $\left\langle d, \dot{d} \right\rangle = 0$. Using both s and d as the first two columns of a rotation matrix function $J : \mathcal{T} \to \mathsf{SO}(3)$ the third column can be uniquely completed as a unit normal to the surface. This way, $\dot{c} = \cos(z) J\mathbf{e}_1$ and $d = J\mathbf{e}_2$. Differentiating the last equality, results in:

$$\dot{d}(t) = \dot{J}(t)\mathbf{e}_2 = J(t) \begin{bmatrix} 0 & z_J(t) & y_J(t) \\ -z_J(t) & 0 & x_J(t) \\ -y_J(t) & -x_J(t) & 0 \end{bmatrix} \mathbf{e}_2 = J(t)(z_J(t)\mathbf{e}_1 - x_J(t)\mathbf{e}_3)$$

where the functions $x_J, y_J, z_J : \mathcal{T} \to \mathbb{R}$ define the anti-symmetric matrix of the tangent vector to a curve defined on $\mathsf{SO}(3)$.

The torsal ruled surface condition implies that $\cos(z) x_J = 0$. Next it will be proven that $x_J = 0$ even when $\cos(z) = 0$. To do so, consider a torsal surface parameterized by the two functions c and d, such that $\dot{c} = 0 \Rightarrow \cos(z) = 0$. Since one is free to translate across the rulings, these functions can be a translation by a non-zero constant f along the rulings of the functions $\hat{c}$ and $\hat{d}$, i.e. $c = \hat{c} + f\hat{d}$ and $d = \hat{d}$ disregarding the normalizing re-parameterization which only affects the derivatives as a scale factor. For this parameterization of the surface $\dot{\hat{c}} = \cos(z) J(t)\mathbf{e}_1 = -f\dot{\hat{d}}$ where the last equality stems from the fact that $\dot{c} = 0$. The torsal ruled condition implies that $\dot{\hat{d}} = J(t)(z_J(t)\mathbf{e}_1 - x_J(t)\mathbf{e}_3)$ but as seen it can only linearly depend on $\mathbf{e}_1$, hence $x_J = 0$. Since $\dot{\hat{d}} = \dot{d}$, this condition propagates to the original $\dot{d}$.

The fact that $\delta = \sin(z)$ forces $z_J = \sin(z)$. Hence, the model so far is

Fact 5.3 (Normalized Torsal Ruled Surface - Differential Model) *Denote the union of $\mathbb{R}$ with plus and minus infinity as $\bar{\mathbb{R}}$. Starting from a set $\mathcal{T} \subset \mathbb{R}$ and two functions $\alpha : \mathcal{T} \to \bar{\mathbb{R}}$ and $\beta : \mathcal{T} \to \bar{\mathbb{R}}$ such that $\alpha(t) < \beta(t)$, define a set $\mathcal{C} = \{(t, v) : t \in \mathcal{T}, v \in]\alpha(t), \beta(t)[\}$.*

*The **differential model of a normalized torsal ruled surface** is defined as the septuple $(\mathcal{T}, \alpha, \beta, z, y, \mathbf{J}_0, \mathbf{c}_0)$ as the image of set $\mathcal{C}$ through the function*

$$r : \mathcal{C} \to \mathbb{R}^3 : r(t, v) = c(t) + vd(t)$$

where $c : \mathcal{T} \subset \mathbb{R} \to \mathbb{R}^3$ is the function obtained by integrating

$$\dot{c} = \cos(z)\ J\mathbf{e}_1, \quad c(0) = \mathbf{c}_0,$$

$J : \mathcal{T} \subset \mathbb{R} \to \mathsf{SO}(3)$ is obtained from the integration of

$$\dot{J} = J \begin{bmatrix} 0 & \sin(z) & y \\ -\sin(z) & 0 & 0 \\ -y & 0 & 0 \end{bmatrix}, \quad J(0) = \mathbf{J}_0$$

and $d : \mathcal{T} \subset \mathbb{R} \to \mathbb{S}(2) \subset \mathbb{R}^3$ is defined as $d = J\mathbf{e}_2$.

Additionally, for every $t \in \mathcal{T}$ the value $-\frac{\cos(z(t))}{\sin(z(t))}$ does not belong to the interval $]\alpha(t), \beta(t)[$ (new immersion condition).

Note that almost all of the previous conditions are now implicitly satisfied, apart from the immersion condition which still needs to be stated. The two initial conditions $\mathbf{c}_0$ and $\mathbf{J}_0$ define an origin and an orientation on the surface.

Up to now, it's not clear why this is a better description than the previous one but the next section provides the answer.

5.3 Isometries

Starting from a function $z : \mathcal{T} \to \mathbb{R}$, and using the initial condition $\mathbf{T}_0 \in \mathsf{SO}(2)$ integrate the function $T : \mathcal{T} \to \mathsf{SO}(2)$ as

$$\dot{T} = T \begin{bmatrix} 0 & \sin(z) \\ -\sin(z) & 0 \end{bmatrix}, \quad T(0) = \mathbf{T}_0.$$

Define $a : \mathcal{T} \to \mathbb{R}^2$ using the initial condition $\mathbf{a}_0 \in \mathbb{R}^2$ and integrate the differential equation

$$\dot{a} = \cos(z)T\mathbf{e}_1, \quad a(0) = \mathbf{a}_0.$$

Define the function $b : \mathcal{T} \to \mathbb{S}(1)$ as $b = J\mathbf{e}_2$. This way, the functions a and b are defined in a similar way as the functions c and d in the previous section. Now it will be shown that if the function z is the same as the function z used in the previous section, a trivial way of defining isometries is obtained.

Consider the function $q : \mathcal{C} \to \mathbb{R}^2$ analogous to r, defined as $q(t, v) = a(t) + vb(t)$ and pullback the metric of $\mathbb{R}^2$ to $\mathcal{C}$ so as to make q a (local) isometry between $\mathcal{C}$ and $\mathbb{R}^2$. To do so, consider any two tangent vectors $\mathbf{v}, \mathbf{w} \in \mathbb{T}_{(t,v)}\mathcal{C}$ and define an inner product as

$$\langle \mathbf{v}, \mathbf{w} \rangle_{\mathcal{C}} = \langle q_*\mathbf{v}, q_*\mathbf{w} \rangle_{\mathbb{R}^2} = \mathbf{v}^T \begin{bmatrix} (\cos(z) + v\sin(z))^2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{w}$$

where the last equation implies the use of orthonormal coordinates on $\mathbb{R}^2$. Consider now another copy of $\mathcal{C}$ (call it $\hat{\mathcal{C}}$) and pullback the metric of the surface $\mathcal{S} \subset \mathbb{R}^3$ so that r is a (local) isometry. The vectors are now $\mathbf{v}, \mathbf{w} \in \mathbb{T}_{(t,v)}\hat{\mathcal{C}}$

$$\langle \mathbf{v}, \mathbf{w} \rangle_{\hat{\mathcal{C}}} = \langle r_*\mathbf{v}, r_*\mathbf{w} \rangle_{\mathbb{R}^3} = \mathbf{v}^T \begin{bmatrix} (\cos(z) + v\sin(z))^2 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{w}$$

Since the metric structure of both copies of $\mathcal{C}$ agree when the same function z is used, the function $id : \mathcal{C} \to \hat{\mathcal{C}}$ that maps points identically, is an isometry. See figure 5.5 for a representation. Note that this map only guarantees a local isometry since as discussed previously, not all torsal ruled surfaces admit a global isometry. Since the discussion is not dependent on the choice of initial conditions $\mathbf{a}_0$ and $\mathbf{J}_0$ they can be forced to be 0 and the identity matrix respectively.

Is figure 5.5 the pre-image of the rulings are labeled e. These are the image of the straight lines parameterized as

$$e_t(v) = a(t) + vb(t)$$

and will play an important part in the next sections. These will also be called **rulings** of the surface.

Figure 5.5 justifies the function equality $\mathcal{I} \circ q = r$. These functions induce the linear functions on the tangent space (known as push-forwards) as

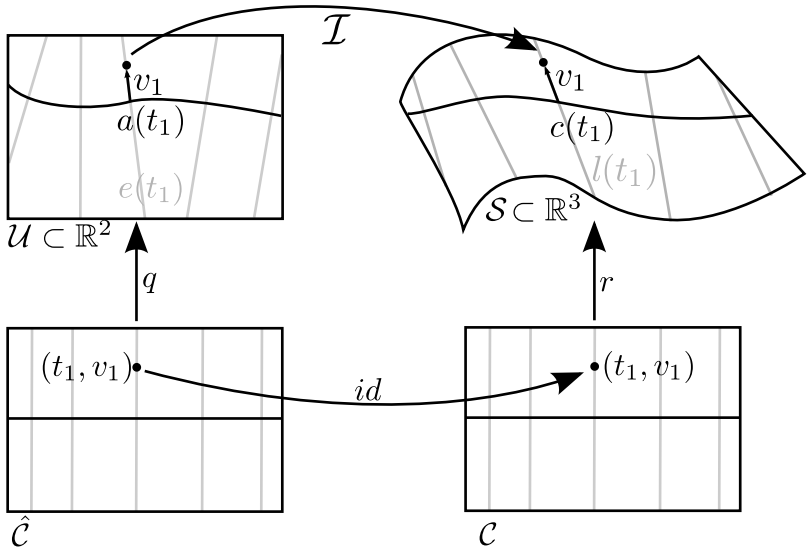


Figure 5.5: The curves a and c induce coordinate systems on $\mathcal{U}$ and $\mathcal{S}$ on which the isometry $\mathcal{I}$ is represented as the identity function.

$$
\begin{aligned}
& & r_{*(t,v)} &= \mathcal{I}_{*q(t,v)} q_{*(t,v)} \\
\iff & & \begin{bmatrix} \dot{c}(t) + v\dot{d}(t) & d(t) \end{bmatrix} &= \mathcal{I}_{*q(t,v)} \begin{bmatrix} \dot{a}(t) + v\dot{b}(t) & b(t) \end{bmatrix} \\
\iff & & \begin{cases} (\cos z(t) + v \sin z(t)) J(t)\mathbf{e}_1 = (\cos z(t) + v \sin z(t)) \mathcal{I}_{*q(t,v)} T(t)\mathbf{e}_1 \\ J(t)\mathbf{e}_2 = \mathcal{I}_{*q(t,v)} T(t)\mathbf{e}_2 \end{cases}
\end{aligned}
$$

Since the immersion condition of the differential model for torsal ruled surfaces states that $\cos z(t) + v \sin z(t) \neq 0$ for all $t \in \mathcal{T}$, this term can be factored out of the first equation resulting in

$$
J(t) \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} = \mathcal{I}_{*q(t,v)} T(t)
$$

or that the first two columns of $J(t)$ satisfy the right side. Remember that the third column of $J(t)$ was simply the normal to the surface and is a function of the first two columns. In the next sections, it saves on the symbol complexity if $\mathbb{R}^2$ is identified with the plane $\{(x, y, z) \in \mathbb{R}^3 : z = 0\}$. This way, $\mathcal{I}_{*q(t,v)}$ can be completed so as to belong to $\mathsf{SO}(3)$ (similarly to what was done to the third column of J) and $T(t)$ can also be seen as a rotation around the z axis in $\mathsf{SO}(3)$. Here, these "extensions" are

written as $\bar{\mathcal{I}}_{*q(t,v)} \in \mathsf{SO}(3)$ and $\bar{T}(t) \in \mathsf{SO}(3)$ respectively. Using this the former equation reads

$$J(t) = \bar{\mathcal{I}}_{*q(t,v)}\bar{T}(t)$$

The first conclusion is that $\bar{\mathcal{I}}_{*q(t,v)} = J(t)\bar{T}^T(t)$ is constant along v. This way, it can be seen as a function of the rulings parameterized by t, and will be written simply as $\bar{\mathcal{I}}_*(t)$.

5.4 The Relation Between the Isometry and the Rotation Matrices

Starting from the equation

$$J = \bar{\mathcal{I}}_*\bar{T}$$

differentiate it:

$$\dot{J} = \dot{\bar{\mathcal{I}}}_*\bar{T} + \bar{\mathcal{I}}_*\dot{\bar{T}} \tag{5.1}$$

$$\iff JK_J = \bar{\mathcal{I}}_*K_I\bar{T} + \bar{\mathcal{I}}_*\bar{T}K_T \tag{5.2}$$

$$\iff JK_J = J\bar{T}^TK_I\bar{T} + JK_T \tag{5.3}$$

$$\iff K_J = \bar{T}^TK_I\bar{T} + K_T \tag{5.4}$$

where the various K that appear are functions of t, representing the tangent vector to the corresponding curve on $\mathsf{SO}(3)$. Parameterizing $\bar{T}$ as

$$T(t) = \begin{bmatrix} c_T(t) & -s_T(t) & 0 \\ s_T(t) & c_T(t) & 0 \\ 0 & 0 & 1 \end{bmatrix},$$

where the functions represent the cosine and sine of a function of t, allows the previous equation to be re-written as

$$\begin{bmatrix} 0 & z_J & y_J \\ -z_J & 0 & 0 \\ -y_J & 0 & 0 \end{bmatrix} = \begin{bmatrix} c_T & s_T & 0 \\ -s_T & c_T & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & z_I & y_I \\ -z_I & 0 & x_I \\ -y_I & -x_I & 0 \end{bmatrix} \begin{bmatrix} c_T & -s_T & 0 \\ s_T & c_T & 0 \\ 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 0 & z_T & 0 \\ -z_T & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

$$\iff \begin{bmatrix} 0 & z_J - z_T & y_J \\ -z_J + z_I & 0 & 0 \\ -y_J & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & z_I & c_T y_I + s_T x_I \\ -z_I & 0 & c_T x_I - s_T y_I \\ -c_T y_I - s_T x_I & -c_T x_I + s_T y_I & 0 \end{bmatrix} \tag{5.6}$$

where the anti-symmetric matrices have also been expanded to their coordinate functions. This yields the relations

$$\begin{aligned} z_J - z_T &= z_I \\ c_T\, y_I + s_T\, x_I &= y_J \\ c_T\, x_I - s_T\, y_I &= 0 \end{aligned} \tag{5.7}$$

the bottom two imply that

$$\begin{aligned} x_I &= s_T\, y_J \\ y_I &= c_T\, y_J \end{aligned} \tag{5.8}$$

And since the last section forced $z_J = z_T$ the first equation implies that $z_I(t)$=0.

From the above discussion one learns that if one considers the curve $\mathcal{I}_*(t) \in \mathsf{SO}(3)$ then the tangent vector to this curve must obey

$$\dot{\mathcal{I}}_* = \mathcal{I}_*\, y_J \underbrace{\begin{bmatrix} 0 & 0 & c_T \\ 0 & 0 & s_T \\ -c_T & -s_T & 0 \end{bmatrix}}_{K_I}$$

where c_T and s_T parameterize the rotation matrix $T(t) \in \mathsf{SO}(2)$.

5.5 Optimization

At the end of chapter 3 the intent of optimizing the cost function in equation 3.8 was stated, but quickly dismissed due to the difficulty posed by the constraint that $\mathcal{I}^k$ must be an isometry. This equation is

here repeated for convenience

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i,k)\in\mathcal{V}} \left\| \mathbf{p}_i^k - s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathcal{I}^k\left(\mathbf{q}_i\right) \right\|_2^2 \\ \text{s.t.} \quad & s^k \in \mathbb{R}^+ \\ & \mathcal{I}^k \text{ isometry} \\ & \mathbf{q}_i \in \mathbb{R} \quad \forall i \end{aligned} \tag{5.9}$$

This chapter removed this difficulty by providing an effective way of representing isometries. The optimization problem is re-written as

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{(i,k)\in\mathcal{V}} \left\| \mathbf{p}_i^k - s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(c^k\left(t_i^k\right) + v_i^k d^k\left(t_i^k\right)\right) \right\|^2 \\ \text{s.t.} \quad & \mathbf{q}_i = a^k(t_i^k) + v_i^k b^k(t_i^k) \\ & \dot{a}^k(t) = \cos(z^k(t)) T^k(t)\mathbf{e}_1 \qquad \dot{c}^k(t) = \cos(z^k(t)) J^k(t)\mathbf{e}_1 \\ & b^k(t) = T^k(t)\mathbf{e}_2 \qquad d^k(t) = J^k(t)\mathbf{e}_2 \\ & \dot{T}^k = T^k \begin{bmatrix} 0 & \sin(z^k) \\ -\sin(z^k) & 0 \end{bmatrix} \qquad \dot{J}^k = J^k \begin{bmatrix} 0 & \sin(z^k) & y^k \\ -\sin(z^k) & 0 & 0 \\ -y^k & 0 & 0 \end{bmatrix} \\ & T^k(0) = \mathbf{T}_0^k \qquad J^k(0) = \mathbf{J}_0^k \\ & a^k(0) = \mathbf{a}_0^k \qquad c^k(0) = \mathbf{c}_0^k \\ & \mathbf{T}_0^k \in \mathsf{SO}(2) \qquad \mathbf{J}_0^k \in \mathsf{SO}(3) \\ & \mathbf{a}_0^k \in \mathbb{R}^2 \qquad \mathbf{c}_0^k \in \mathbb{R}^3 \\ & t_i^k \in \mathcal{T}^k \\ & v_i^k \in \left]\alpha^k(t_i^k),\ \beta^k(t_i^k)\right[\\ & z^k : \mathcal{T}^k \mapsto \mathbb{R} \\ & y^k : \mathcal{T}^k \mapsto \mathbb{R} \end{aligned}$$

At first glance the problem seems overwhelming, particularly since in involves an optimization problem of infinite dimension on the space of functions (z^k and y^k). But it does provide a way to represent the problem which was not previously available.

Since no reliable information exists about the set $\mathcal{T}^k$ or $\mathcal{C}^k$, these will be relaxed to be $\mathbb{R}$ and $\mathbb{R}^2$

respectively. This implies that the immersion condition is not being explicitelly enforced, but knowing that the data comes from real embedded isometric surfaces if the optimization problem converges to the true solution the condition will be satisfied. So as long as the algorithm is working as expected, the immersion condition should be automatically satisfied, otherwise the solution is meaningless anyway.

Although there are people working on optimization algorithms on the space of functions, at this time a simplified approach is taken. Instead of considering any function, the functions z^k and y^k will instead be approximated by piecewise constant functions, discontinuous at half the distance between each data point. The new symbols $z_i^k = z^k(t_i^k)$ and $y_i^k = y^k(t_i^k)$ are introduced to represent the functions at the sampled values. Similarly, define $\mathbf{J}_i^k$, $\mathbf{T}_i^k$, $\mathbf{c}_i^k$, $\mathbf{d}_i^k$, $\mathbf{a}_i^k$ and $\mathbf{b}_i^k$.

This way, the differential equations on the rotation matrices in the problem are simple to integrate. Since it will be useful later on, the symbols $\mathbf{J}_{i-1/2}^k$ and $\mathbf{T}_{i-1/2}^k$ are also defined, which correspond to the integration half way between two sample points

$$\mathbf{J}_{i-1/2}^k = \mathbf{J}_{i-1}^k \exp\left(\frac{t_i^k - t_{i-1}^k}{2}\begin{bmatrix} 0 & \sin(z_{i-1}^k) & y_{i-1}^k \\ -\sin(z_{i-1}^k) & 0 & 0 \\ -y_{i-1}^k & 0 & 0 \end{bmatrix}\right)$$

$$\mathbf{J}_i^k = \mathbf{J}_{i-1/2}^k \exp\left(\frac{t_i^k - t_{i-1}^k}{2}\begin{bmatrix} 0 & \sin(z_i^k) & y_i^k \\ -\sin(z_i^k) & 0 & 0 \\ -y_i^k & 0 & 0 \end{bmatrix}\right)$$

$$\mathbf{T}_{i-1/2}^k = \mathbf{T}_{i-1}^k \exp\left(\frac{t_i^k - t_{i-1}^k}{2}\begin{bmatrix} 0 & \sin(z_{i-1}^k) \\ -\sin(z_{i-1}^k) & 0 \end{bmatrix}\right)$$

$$\mathbf{T}_i^k = \mathbf{T}_{i-1/2}^k \exp\left(\frac{t_i^k - t_{i-1}^k}{2}\begin{bmatrix} 0 & \sin(z_i^k) \\ -\sin(z_i^k) & 0 \end{bmatrix}\right)$$

The matrix exponential functions for these cases are easy to compute, using the Rodrigues rotation formula for the 3 dimensional case and the fact that $\mathsf{SO}(2)$ is group isomorphic to the circle, hence it results in a simple rotation of an angle equal to the anti-symmetric argument.

To integrate the second set of differential equations note that the velocity vector $\dot{c}$ is orthogonal to the acceleration vector $\ddot{c}$ and both are of constant magnitude whenever z and y are constant. Taking the cross product between the two, one also discovers that the result does not depend on the rotation angle, thus the movement is planar. This implies that the curve c is piecewise circular arches. The

same is true for the curve a. In any circular path, the angular velocity at which it is run is given by $\omega = \|\dot{c}\| \,/\, \|\ddot{c}\|$ and the center of the path is $\|\dot{c}\| \,/\omega$ in the direction of the acceleration vector. Thus, in this particular case $\omega = \sqrt{\sin(z)^2 + y^2}$ and the center point of the trajectory is $c + \ddot{c}/(\sin(z)^2 + y^2)$. Thus, using all that was said, the following function is proposed as the integration of a section of $\dot{c}$, starting at $t = 0$ with initial conditions $c(0) = 0$ and $\mathbf{J}_0$, when the functions y^k and z^k are constant:

$$i_c(t; \mathbf{J}, y, z) = \frac{\cos(z)}{\sin(z)^2 + y^2}\ \mathbf{J} \begin{bmatrix} 0 \\ -\sin(z) \\ -y \end{bmatrix} + \frac{\cos(z)}{\sqrt{\sin(z)^2 + y^2}}\ \mathbf{J} \begin{bmatrix} \sin(\sqrt{\sin(z)^2 + y^2}\ t) \\ \frac{\sin(z)}{\sqrt{\sin(z)^2+y^2}} \cos(\sqrt{\sin(z)^2 + y^2}\ t) \\ \frac{y}{\sqrt{\sin(z)^2+y^2}} \cos(\sqrt{\sin(z)^2 + y^2}\ t) \end{bmatrix}$$

Note that the important limit $\sin(z)^2 + y^2 \to 0$ where the function degenerates to

$$i_c(t; \mathbf{J}, y, z) = t\ \mathbf{J}\ \mathbf{e}_1$$

This way, to integrate the path when y^k and z^k are piecewise constant, one gets

$$\mathbf{c}_i^k = \mathbf{c}_{i-1}^k + i_c((t_i - t_{i-1})/2; \mathbf{J}_{i-1}^k, y_{i-i}^k, z_{i-1}^k) + i_c((t_i - t_{i-1})/2; \mathbf{J}_{i-1/2}^k, y_i^k, z_i^k)$$

In a much simpler way, the integration formula for a^k is given by

$$i_a(t; \mathbf{T}, z) = -\frac{\cos(z)}{\sin(z)}\ \mathbf{T} \begin{bmatrix} 0 \\ 1 \end{bmatrix} + \frac{\cos(z)}{\sin(z)}\ \mathbf{T} \begin{bmatrix} \sin(\sin(z)\ t) \\ \cos(\sin(z)\ t) \end{bmatrix}$$

and likewise, the important limit $\sin(z) \to 0$ where the function again degenerates to a straight line

$$i_a(t; \mathbf{T}, z) = t\ \mathbf{T}\ \mathbf{e}_1$$

The integrated path is thus

$$\mathbf{a}_i^k = \mathbf{a}_{i-1}^k + i_a((t_i - t_{i-1})/2; \mathbf{T}_{i-1}^k, z_{i-1}^k) + i_a((t_i - t_{i-1})/2; \mathbf{T}_{i-1/2}^k, z_i^k)$$

To save on the symbol complexity, all these equations will be hidden behind the phrase "$\mathcal{I}^k$ isom-

etry". Thus the optimization problem is now

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,k)\in\mathcal{V}} \left\| s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(\mathbf{c}_i^k + v_i^k \mathbf{d}_i^k\right) - \mathbf{p}_i^k \right\|^2 \\ \text{s.t.} \quad & \mathbf{q}_i = \mathbf{a}_i^k + v_i^k \mathbf{b}_i^k \\ & \mathcal{I}^k \text{ isometry} \end{aligned}$$

Here the points $\mathbf{q}_i$ may be assumed to be variables of the problem or known. In the later case the optimization problem is actually an extension of what was discussed in chapter 4 with respect to pose estimation. Either way, the usual technique of using a penalty function where a parameter $\mu \in \mathbb{R}$ is progressively increased so as to force the constraint to be obeyed may be used to solve the constrained optimization as

$$\begin{aligned} \text{minimize} \quad & \sum_{(i,k)\in\mathcal{V}} \left\| s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(\mathbf{c}_i^k + v_i^k \mathbf{d}_i^k\right) - \mathbf{p}_i^k \right\|^2 + \mu \left\| \mathbf{a}_i^k + v_i^k \mathbf{b}_i^k - \mathbf{q}_i \right\|^2 \\ \text{s.t.} \quad & \mathcal{I}^k \text{ isometry} \end{aligned}$$

Note that the remaining constraints may be plugged directely into the cost function meaning that the problem is now unconstrained. This problem is solvable using gradient methods and the author obtained good results using the Levenberg–Marquardt algorithm for least squares optimization. In the case where $\mathbf{q}_i$ are known, the problem decouples for each image k. In the case where they are variables, the problem may be computationally simplified by a standard "decoupling" technique by writting

$$\begin{aligned} \text{minimize} \quad & \left(\begin{aligned} \text{minimize} \quad & \sum_{(i,k)\in\mathcal{V}} \left\| s^k \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \left(\mathbf{c}_i^k + v_i^k \mathbf{d}_i^k\right) - \mathbf{p}_i^k \right\|^2 + \mu \left\| \mathbf{a}_i^k + v_i^k \mathbf{b}_i^k - \mathbf{q}_i \right\|^2 \\ \text{s.t.} \quad & \mathcal{I}^k \text{ isometry} \end{aligned} \right) \\ \text{s.t.} \quad & \mathbf{q}_i \in \mathbb{R}^2 \end{aligned}$$

where the inner problems are solved for fixed $\mathbf{q}_i$ and then an iteration of the exterior problem is run. These steps are cycled until a convergence criterion is met.

There is still a significant problem that must be addressed before the optimization can be carried

out, relating to the initialization of the gradient method. The previous chapters provided a way for an initial solution to be obtained, but unfortunately the parameters obtained are not directly convertible to the parameters needed. The next section explains how to convert the previously obtained data into the new description.

5.6 Numeric Initialization

In the previous chapters a way of inferring the flattened surface, represented by the points $\mathbf{q}_i \in \mathcal{Q} \subset \mathbb{R}^2$ and a local estimate of $\mathcal{I}_{*\mathbf{q}_i}$, was presented. These estimates are going to be used to initialize the continuous surface parameters introduced in this chapter.

Numerically, suppose that one knows the value of $\mathcal{I}_*$ at two close points $\mathbf{q}_1, \mathbf{q}_2 \in \mathbb{R}^2$ which lie on the rulings parameterized by t_1 and t_2: $\mathbf{q}_1 \in e_l(t_1)$ and $\mathbf{q}_2 \in e_l(t_2)$. A first order approximation for $\mathcal{I}_*(t)$ is

$$\mathcal{I}_*(t_1 + h) \approx \mathcal{I}_*(t_1) \exp^{h(\mathcal{I}_*(t_1))^T \dot{\mathcal{I}}_*(t1)} = \mathcal{I}_*(t_1) \exp^{hK_I(t_1)}$$

Considering h such that $t_2 = t_1 + h$ and rearranging the equation:

$$\begin{aligned} \log\left(\left(\mathcal{I}_*(t_1)\right)^T \mathcal{I}_*(t_2)\right) &\approx (t_2 - t_1) K_I(t_1) \\ &= (t_2 - t_1) y_J(t_1) \begin{bmatrix} 0 & 0 & c_T(t_1) \\ 0 & 0 & s_T(t_1) \\ -c_T(t_1) & -s_T(t_1) & 0 \end{bmatrix} \end{aligned}$$

The first thing to notice is that the matrix $T(t_1)$ can be obtained up to a sign, even if $(t_2 - t_1)y_J(t_1)$ is unknown, by normalizing the last column of $K_I(t_1)$ to unit length. This is enough to infer the ruling $e(t_1)$ that passes through the point.

5.6.1 Greedily Inferring the Rulings

Besides the first order approximation used in the last equation, there is also the problem introduced by noise in the data. So the ruling that passes through a given point $\mathbf{q}_i$, can be better approximated from

all the neighboring values by solving the bilinear system of equations

$$\begin{aligned} h_1\mathbf{K}_I &= \log((\mathcal{I}_{*i})^T\mathcal{I}_{*1_i}) \\ h_2\mathbf{K}_I &= \log((\mathcal{I}_{*i})^T\mathcal{I}_{*2_i}) \\ h_3\mathbf{K}_I &= \log((\mathcal{I}_{*i})^T\mathcal{I}_{*3_i}) \\ \vdots \quad & \quad \vdots \end{aligned}$$

where the notation j_i was introduced in the previous chapters and means the j'th neighbor of point i. Since only two independent parameters are available in $\mathbf{K}_I$, this system can be solved in a least squares sense, as a rank-1 factorization on the entries $(1,3)$ and $(2,3)$ of the matrices. The solution represents a line through each of the points, which approximates the rulings $e(t_i)$ of the surface. Note that since neighboring points are local to each point, there is still no consistency between non-neighboring points on the same ruling.

5.6.2 Smoothing the Rulings to Obey the Global Constraint

To guarantee consistency between non-neighboring points a piecewise cone approximation is used. Under this model, nearby rulings must intersect at a common point, which might be infinity allowing for parallel lines. Suppose that a nominal ruling is available and consider all the points within a certain distance to it. The previous discussion provided a line direction through each of these points and one now wishes to smooth these so that they intersect at a common point. This is known in projective geometry as a pencil of lines and can be estimated as a rank projection much the same way as the best line that passes through a cloud of points (which is the dual problem in projective geometry). This process assigns smoothed rulings to neighboring points, and the process can be called recursively until all points have a smoothed ruling attributed to them. The first ruling can be found by searching the space of lines that pass through the center of the constellation of points. Once all the rulings have been found, one can consider circular arches that intersect them orthogonally, yielding the image of the curve a.

This curve must then parameterized so as to obey the constraint $\|\dot{a}\|^2 = \cos^2 z$ and $\left\|\dot{b}\right\|^2 = \sin^2 z$, which depends on the radius of the circular arch. This step yields the value of the parameters z_i^k and t_i^k. Now that t_i^k are available, the h_i found in section 5.6.1 provide the final parameters y_i^k.

Two examples of point clouds converted this way can be seen in figure 5.6. After the optimization algorithm is applied, the results obtained are show in figure 5.7. For the image of the half cylinder, the

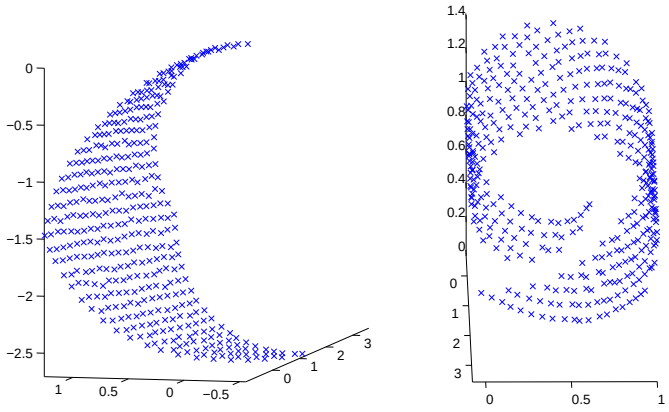


Figure 5.6: First estimates of the surfaces converted to the continuous model. An image wrapped around a cylinder and the swiss roll are shown as the image of $c^k(t_i^k) + v_i^k d^k(t_i^k)$. The noise level on the original images was $\sigma_{noise} = 0.01$.

before and after mean squared errors when compared to the ground truth 3D surface are respectively 0.1014 and 0.0003. For the image of the swiss roll, the mean squared errors are respectively 0.1100 and 0.0005. Both of these confirm that applying the intrinsically correct model described in this chapter provides better reconstructions.

Figure 5.8 presents the reconstruction of the points seen on the real world image using the differential model presented in this chapter.

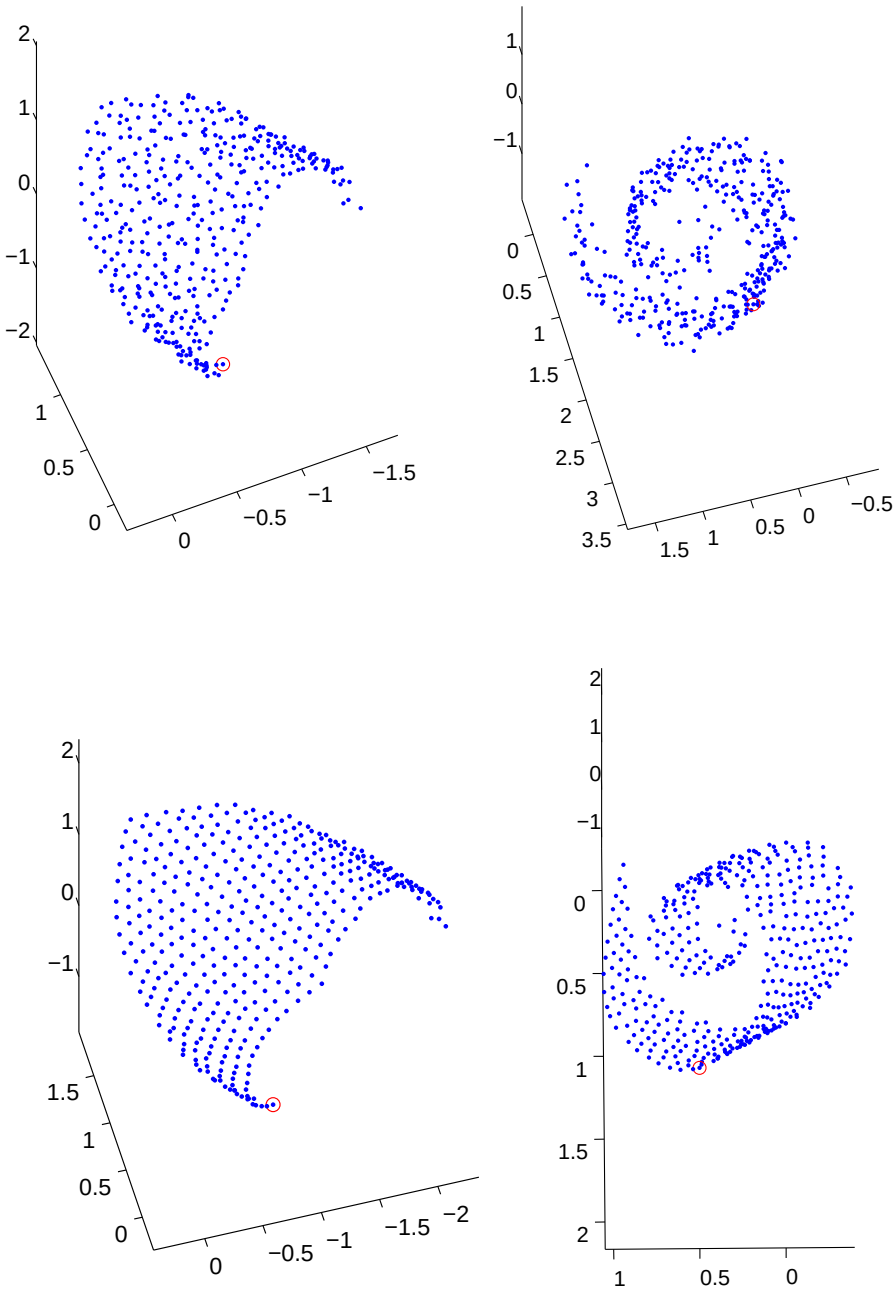


Figure 5.7: The results obtained in chapter 4 are shown above, which are used to initialize the algorithm provided in this chapter, producing the results shown below. An image wrapped around a cylinder and the swiss roll are shown as the image of $c^k(t_i^k) + v_i^k d^k(t_i^k)$. The noise level on the original images was $\sigma_{noise} = 0.01$.

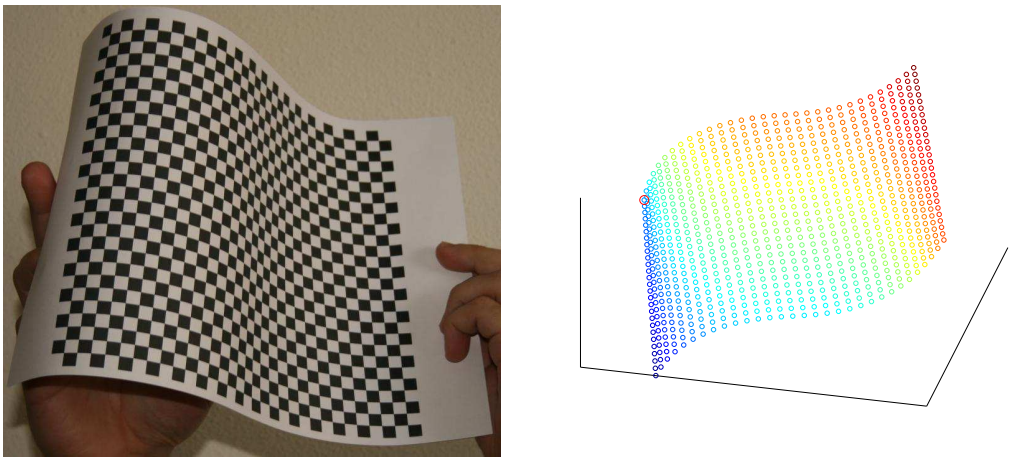


Figure 5.8: Pose estimation using the differential model description. On the left the observed image, on the right the reconstruction of the points described as a differential model.

Chapter 6

Conclusion

This thesis proposed several new techniques relating to representation and reconstruction of surfaces with the distinct property of having everywhere zero curvature. The methodology involves sequentially solving several problems, starting from matched features in different 2D images and culminating in first a point-wise characterization of the unfolded surface and later a functional description of the surface and scene. The cameras are assumed to be scale orthographic and are not previously calibrated.

Two separate techniques are described: the first based on rank factorization of matrices relying solely on local properties of the observed surfaces and operating solely on the observed point clouds; the second modeling the surface as a set of differential equations on which optimization may be performed which allows for all non-local properties to be satisfied implicitly.

Although the problem of reconstructing the unfolded surface has an almost unique solution, in the sense that it is unique up to scale, rotation and translation, the problem of obtaining the embedding of the surface as seen on an image does not. This is not a limitation of the presented work but a limitation of the problem itself which can not be solved unless further information is provided.

6.1 Limitations and Future Work

The greatest limitation of the proposed methodology is its reliance on matched image features. Finding the correct correspondence between points in different images is a hard enough problem when the scene is known to be rigid. When considering non-rigid scenes with possible occlusions, as is the case in this thesis, obtaining the needed input data automatically is a challenge. Also, a reliable way of detecting outlier data and discarding it should be implemented, or change the methodology to be robust to these matching errors.

A second limitation is the sequential nature of the method, both inside the solutions provided in chapters 3 and 4 and the sequential connection between chapters 3, 4 and 5. Changing the sequential methods in chapters 3 and 4 would probably imply the development of a completely different algorithm, but if a way to initialize the optimization problem in chapter 5 were found which did not rely on the previous knowledge of a solution it would effectively decouple the chain and allow this algorithm to stand by itself.

Due to the author's limited knowledge of infinite dimensional functional optimization, the optimization parameters in chapter 5 were simplified to turn the problem into the finite dimensional domain. Here two possible improvements are possible by either improving on the way the discrete parameters are chosen and integrated or performing the optimization directly in the space of functions.

The pose estimation problem presented in chapter 4 which also affects the embedding in chapter 5 has a natural ambiguity due to the camera model not providing any information on depth. In this thesis not much attention was payed to it, but it is the author's belief that using second order information (smoothness of the bending rate) near the points where the surface is almost parallel to the camera's projection plane should allow for more embedding information. Also, if stream information is available (i.e. video images) using the dynamics of the surface movement should also improve both the reconstructions and the camera ambiguities.

Finally, the presented algorithms are all batch algorithms which assume all information is available at once. Since the number of observed images influences the noise of the reconstruction, it seems that a video sequence of a waving surface would be better to minimize reconstruction noise. Unfortunately, the amount of data that needs to be held makes this unviable for large datasets. Ideally, a way to update the reconstructed data at each frame without relying on the past frames would be optimal.

Appendix A

Sub-Stiefel Set

Due to their special importance to this work, a certain type of matrices here named **sub-Stiefel** matrices ($\mathbb{SS}$) will be exhaustively described in this section. These matrices are obtained from the $\mathsf{SO}(3)$ matrices by truncating the last column and the last row, hence

$$\mathbb{SS} = \left\{ \mathbf{S} \in \mathsf{M}^{2\times 2} : \begin{bmatrix} \mathbf{S} & * \\ * & * \end{bmatrix} \in \mathsf{SO}(3) \right\} = \left\{ \mathbf{S} \in \mathsf{M}^{2\times 2} : \begin{bmatrix} \mathbf{S} \\ * \end{bmatrix} \in \mathsf{O}(3,2) \right\}.$$

Note that either representation is equivalent.

Theorem A.1 *If $\mathbf{S} \in \mathbb{SS}$ is a sub-Stiefel matrix, then so is the matrix generated by pre and post multiplying rotation matrices $\mathbf{R}_1, \mathbf{R}_2 \in \mathsf{SO}(2)$*

Proof This stems from the fact that $\mathsf{SO}(2)$ is canonically embedded in $\mathsf{SO}(3)$ as

$$\mathsf{SO}(2) \simeq \left\{ \begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathsf{SO}(3) : \mathbf{R} \in \mathsf{SO}(2) \right\}.$$

$\mathbf{S} \in \mathbb{SS}$ means that there is a rotation matrix $\begin{bmatrix} \mathbf{S} & * \\ * & * \end{bmatrix} \in \mathsf{SO}(3)$. By embedding $\mathbf{R}_1, \mathbf{R}_2 \in \mathsf{SO}(2)$ in $\mathsf{SO}(3)$ and by pre and post multiplication one obtains the matrix $\begin{bmatrix} \mathbf{R}_1\mathbf{S}\mathbf{R}_2 & * \\ * & * \end{bmatrix} \in \mathsf{SO}(3)$ (ignoring the stared parts), which proves the statement. ∎

The operation of matrix truncation has a well known effect on the singular values. In [15] the following theorems describe this operation for symmetric matrices:

Fact A.2 *Let $\mathbf{A} \in \mathbb{R}^{n\times n}$ be a given Hermitian matrix, let $\mathbf{y} \in Cn$ be a given vector, and let $a \in \mathbb{R}$ be a given real number. Let $\hat{\mathbf{A}} \in \mathbb{R}^{(n+1)\times(n+1)}$ be a Hermitian matrix obtained by bordering $\mathbf{A}$ with $\mathbf{y}$ and a as follows: $\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{A} & \mathbf{y} \\ \mathbf{y}^* & a \end{bmatrix}$. Let the eigenvalues of $\mathbf{A}$ and $\hat{\mathbf{A}}$ be denoted by $\{\lambda_i\}_1^n$ and $\left\{\hat{\lambda}_i\right\}_1^{n+1}$, respectively, and assume that they have been arranged in increasing order $\lambda_1 \leq \cdots \leq \lambda_n$ and $\hat{\lambda}_1 \leq \cdots \leq \hat{\lambda}_{n+1}$. Then*

$$\hat{\lambda}_1 \leq \lambda_1 \leq \hat{\lambda}_2 \leq \lambda_2 \leq \cdots \leq \hat{\lambda}_n \leq \lambda_n \leq \hat{\lambda}_{n+1}$$

Proof See [15] Theorem 4.3.8, page 185. ∎

The "converse" is also true:

Fact A.3 *Let $\{\lambda_i\}_1^n$ and $\left\{\hat{\lambda}_i\right\}_1^{n+1}$ be two given sequences of real numbers such that $\hat{\lambda}_1 \leq \lambda_1 \leq \hat{\lambda}_2 \leq \lambda_2 \leq \cdots \leq \hat{\lambda}_n \leq \lambda_n \leq \hat{\lambda}_{n+1}$, then there exists a real vector $\mathbf{y} \in \mathbb{R}^n$ and $a \in \mathbb{R}$ such that $\left\{\hat{\lambda}_i\right\}_1^{n+1}$ is the set of eigenvalues of the real symmetric matrix $\hat{\mathbf{A}} = \begin{bmatrix} \mathbf{\Lambda} & \mathbf{y} \\ \mathbf{y}^T & a \end{bmatrix} \in \mathsf{M}^{(n+1)\times(n+1)}$, where $\mathbf{\Lambda}$ is the diagonal matrix with entries λ_i.*

Proof See [15] Theorem 4.3.10, page 186. ∎

Note that this result extends to the case where $\mathbf{\Lambda}$ is substituted by a symmetric matrix $\mathbf{A} \in \mathsf{M}^{n\times n}$ with eigenvalues $\{\lambda_i\}_1^n$ by considering an eigenvalue decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{U} \in \mathsf{SO}(n)$. Note that $\mathbf{U} \in \mathsf{SO}(n)$ implies $\hat{\mathbf{U}} = \begin{bmatrix} \mathbf{U} & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \in \mathsf{SO}(n+1)$, thus:

$$\operatorname{eig}\left(\hat{\mathbf{A}}\right) = \operatorname{eig}\left(\begin{bmatrix} \mathbf{\Lambda} & \mathbf{y} \\ \mathbf{y}^T & a \end{bmatrix}\right) = \left\{\hat{\lambda}_i\right\}$$

$$\iff \operatorname{eig}\left(\hat{\mathbf{U}}\begin{bmatrix} \mathbf{\Lambda} & \mathbf{y} \\ \mathbf{y}^T & a \end{bmatrix}\hat{\mathbf{U}}^T\right) = \operatorname{eig}\left(\begin{bmatrix} \mathbf{A} & \mathbf{U}\mathbf{y} \\ \mathbf{y}^T\mathbf{U}^T & a \end{bmatrix}\right) = \left\{\hat{\lambda}_i\right\}$$

These are usually known as the Cauchy Interlacing Theorems. In the case where the matrix is not hermitian, a similar theorem applies to the singular values by noting that the squares of the singular values of a matrix $\mathbf{A} \in \mathsf{M}^{n\times m}$ are the eigenvalues of $\mathbf{A}^T\mathbf{A}$ to which the previous result can be applied:

Fact A.4 *Let $\mathbf{A} \in \mathsf{M}^{m\times n}$ be a given matrix and let $\hat{\mathbf{A}}$ be the matrix obtained by deleting any one row from $\mathbf{A}$. Let $\{\sigma_i\}_1^m$ denote the singular values of $\mathbf{A}$ and let $\{\hat{\sigma}_i\}_1^n$ denote the singular values of $\hat{\mathbf{A}}$, both arranged in non-increasing order.*

1. *If $n \geq m$, then*
$$\sigma_1 \geq \hat{\sigma}_1 \geq \sigma_2 \geq \hat{\sigma}_2 \geq \cdots \geq \hat{\sigma}_{m-1} \geq \sigma_m \geq 0$$
2. *If $n < m$, then*
$$\sigma_1 \geq \hat{\sigma}_1 \geq \sigma_2 \geq \hat{\sigma}_2 \geq \cdots \geq \sigma_n \geq \hat{\sigma}_n$$

Proof See [15] Theorem 7.3.9, page 419. ∎

Using theorem A.4, the sub-Stiefel set can be characterized as the set of 2×2 matrices with the largest singular value equal to 1:

Theorem A.5 *The set of sub-Stiefel matrices is given by*

$$\mathbb{SS} = \left\{\mathbf{S} \in \mathsf{M}^{2\times 2} : \sigma_{\max}(\mathbf{S}) = 1\right\}.$$

Proof The proof is broken into 2 parts:

- $\mathbb{SS} \subset \left\{\mathbf{S} \in \mathsf{M}^{2\times 2} : \sigma_{\max}(\mathbf{S}) = 1\right\}$:
  Let
  $$\mathbf{R} = \begin{bmatrix} \mathbf{S} & \mathbf{r} \\ \mathbf{s}^T & u \end{bmatrix} \in \mathsf{SO}(3)$$
  where $\mathbf{S}$ is a sub-Stiefel matrix. Then the matrix
  $$\mathbf{Q} = \begin{bmatrix} \mathbf{S} \\ \mathbf{s}^T \end{bmatrix}$$
  is a Stiefel matrix, hence its singular values are $\{1, 1, 0\}$. Therefore theorem A.4 applies directly, proving that the maximum singular value of $\mathbf{S}$ is 1.
- $\mathbb{SS} \supset \left\{\mathbf{S} \in \mathsf{M}^{2\times 2} : \sigma_{\max}(\mathbf{S}) = 1\right\}$:
  Let $\mathbf{S} \in \mathsf{M}^{2\times 2}$ be a matrix whose maximum singular value is 1 and the other is λ. Then the eigenvalues of $\mathbf{S}\mathbf{S}^T$ are 1 and λ^2. By theorem A.3, there is a vector $\mathbf{y}$ and a scalar $a \in \mathbb{R}$ such that
  $$\operatorname{eig}\left(\begin{bmatrix} \mathbf{S}\mathbf{S}^T & \mathbf{y} \\ \mathbf{y}^T & a \end{bmatrix}\right) = \{1, 1, 0\}$$

Since the matrix is symmetric and positive semi-definite, there is a matrix $\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_1 \\ \mathbf{q}^T \end{bmatrix} \in \mathsf{M}^{3\times 2}$ with singular values equal to $\{1, 1, 0\}$ (Stiefel matrix) such that

$$\mathbf{Q}\mathbf{Q}^T = \begin{bmatrix} \mathbf{S}\mathbf{S}^T & \mathbf{y} \\ \mathbf{y}^T & a \end{bmatrix},$$

in particular $\mathbf{Q}_1\mathbf{Q}_1^T = \mathbf{S}\mathbf{S}^T$ which implies that there is a matrix $\mathbf{R} \in \mathsf{SO}(2)$ such that $\mathbf{S} = \mathbf{Q}_1\mathbf{R}$. Since $\mathbf{Q}_1$ is sub-Stiefel by theorem A.1 $\mathbf{S}$ is sub-Stiefel.

Thus $\mathbb{SS} = \left\{\mathbf{S} \in \mathbb{R}^{2\times 2} : \sigma_{\max}(\mathbf{S}) = 1\right\}$. ∎

Note that this is an intrinsic characterization of the sub-Stiefel set, not dependent on the existence of a rotation (or Stiefel) matrix. It also leads to some interesting equalities:

Theorem A.6 *Let* $\mathbf{S} \in \mathbb{SS}$*, then*

$$\sigma_{\min}^2 = \det(\mathbf{S})^2 = \|\mathbf{S}\|^2 - 1$$

Proof The proof follows from the singular value relations $\det^2(\mathbf{S}) = \sigma_{\max}^2(\mathbf{S})\sigma_{\min}^2(\mathbf{S})$ and $\|\mathbf{S}\|^2 = \sigma_{\max}^2(\mathbf{S}) + \sigma_{\min}^2(\mathbf{S})$ by imposing $\sigma_{\max}(\mathbf{S}) = 1$. ∎

This last theorem, although not of great importance by itself, serves to characterize the sub-Stiefel set as a subset of an algebraic variety:

Theorem A.7 *Define the set* $\mathcal{A} = \left\{\mathbf{A} \in \mathbb{R}^{2\times 2} : \|\mathbf{A}\|^2 - \det(\mathbf{A})^2 - 1 = 0,\ \|\mathbf{A}\|^2 \leq 2\right\}$*, then* $\mathcal{A} = \mathbb{SS}$.

Proof The proof is split into two parts:

1. $\mathbb{SS} \subset \mathcal{A}$:

   Let $\mathbf{S} \in \mathbb{SS}$, then $\sigma_{\max}(\mathbf{S}) = 1$ which trivially satisfies the equality condition $\|\mathbf{S}\|^2 - \det(\mathbf{S})^2 - 1 = 0$:

   $$\sigma_{\max}^2(\mathbf{S}) + \sigma_{\min}^2(\mathbf{S}) - \sigma_{\max}^2(\mathbf{S})\sigma_{\min}^2(\mathbf{S}) - 1 = \sigma_{\min}^2(\mathbf{S}) - \sigma_{\min}^2(\mathbf{S}) = 0$$

   and since $\sigma_{\min}^2(\mathbf{S}) \leq \sigma_{\max}^2(\mathbf{S}) = 1$ the inequality constraint is also satisfied:

   $$\|\mathbf{S}\| = \sigma_{\min}^2(\mathbf{S}) + \sigma_{\max}^2(\mathbf{S}) \leq 1 + 1 = 2$$

2. $\mathcal{A} \subset \mathbb{SS}$:

   Let $\mathbf{A} \in \mathcal{A}$, then the equality $\|\mathbf{A}\|^2 - \det(\mathbf{A})^2 - 1 = 0$ holds. Using singular values, this can be written as

$$\sigma_{\max}^2(\mathbf{A}) + \sigma_{\min}^2(\mathbf{A}) - \sigma_{\max}^2(\mathbf{A})\sigma_{\min}^2(\mathbf{A}) - 1 = 0$$
$$\Longrightarrow \sigma_{\max}^2(\mathbf{A}) = \frac{1 - \sigma_{\min}^2(\mathbf{A})}{1 - \sigma_{\min}^2(\mathbf{A})}$$

   This implies that either $\sigma_{\max}^2(\mathbf{A}) = 1$ or $\sigma_{\min}^2(\mathbf{A}) = 1$. The inequality constraint $\|\mathbf{A}\|^2 = \sigma_{\max}^2(\mathbf{A}) + \sigma_{\min}^2(\mathbf{A}) \leq 2$ imposes that $\sigma_{\min}^2(\mathbf{A}) \leq 1$. If $\sigma_{\min}^2(\mathbf{A}) = 1$ then the inequality constraint forces that $1 = \sigma_{\min}^2(\mathbf{A}) \leq \sigma_{\max}^2(\mathbf{A}) \leq 1$. Either way, $\sigma_{\max}(\mathbf{A}) = 1$.

Hence $\mathbb{SS} = \mathcal{A}$. ∎

Hence, given a general matrix $\mathbf{A} = \begin{bmatrix} a & c \\ b & d \end{bmatrix}$ the implicit condition that must be satisfied is $f(\mathbf{A}) = a^2+b^2+c^2+d^2-a^2d^2+2abcd+c^2b^2-1 = 0$. A question that arises is if this set is a smooth manifold. Since an implicit condition is available, checking for smoothness resumes itself to guaranteeing that the rank of the differential map is 1. Hence the set is not smooth if the derivatives

$$\frac{\partial f}{\partial a} = 2a - 2ad^2 + 2bcd$$
$$\frac{\partial f}{\partial b} = 2b - 2bc^2 + 2acd$$
$$\frac{\partial f}{\partial c} = 2c - 2cb^2 + 2abd$$
$$\frac{\partial f}{\partial d} = 2d - 2da^2 + 2abc$$

are simultaneously zero (inside the allowed domain). Unfortunately the next theorem states that the set is not smooth at certain well characterized points.

Theorem A.8 *The set $\mathbb{SS}$ is not a smooth sub-manifold of $\mathsf{M}^{2\times 2}$ only at the points $\mathbf{A} \in \mathbb{SS}$ where $\mathbf{A} \in \mathsf{O}(2)$.*

Proof To check where the derivatives are simultaneously zero, the following system must be solved

$$a = (ad - bc)d \qquad b = -(ad - bc)c \qquad c = -(ad - bc)b \qquad d = (ad - bc)a \tag{A.1}$$

substituting the first in the fourth the equation $d = (ad - bc)^2 d$ is obtained, meaning that either $(ad - bc)^2 = 1$ or $d = 0$. Similarly, by substituting the fourth in the first, the second in the third

and the third in the second, the conclusion is that either the squared determinant is 1 or the matrix is zero. Since the zero matrix is not in $\mathbb{SS}$, the only possible case is $(ad-bc)^2 = 1$. In $\mathbb{SS}$ the maximum singular value is already unitary, hence this condition states that the minimum singular value must also be 1 implying that it must be a rotation matrix. The remainder of the proof consists in verifying that all rotation matrices obey the equations (A.1).

1. Case $\mathbf{A} \in \mathsf{SO}(2)$:
   Using the $\mathsf{SO}(2)$ parameterization $\mathbf{A} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$ it is clear that the matrix entries satisfy equations (A.1) since $(ad-bc) = 1$

2. Case $\mathbf{A} \in \mathsf{O}(2) - \mathsf{SO}(2)$:
   Using the parameterization $\mathbf{A} = \begin{bmatrix} \sin(\theta) & \cos(\theta) \\ \cos(\theta) & -\sin(\theta) \end{bmatrix}$ it is clear that the matrix entries satisfy the equations (A.1) since $(ad-bc) = -1$

Hence $\mathbb{SS}$ is smooth except at the points in $\mathsf{O}(2)$. ∎

These orthogonal matrices are the limits of the sub-Stiefel set as the next theorem shows.

Theorem A.9 *Let*

$$\mathcal{C} = \{\alpha\mathbf{R} + (1-\alpha)\mathbf{Q} : \alpha \in [0,1], \mathbf{R} \in \mathsf{SO}(2), \mathbf{Q} \in \mathsf{O}(2) - \mathsf{SO}(2)\}.$$

Then $\mathbb{SS} = \mathcal{C}$.

Proof The proof is broken into 2 parts:

- $\mathbb{SS} \subset \mathcal{C}$:
  Let $\mathbf{S} \in \mathbb{SS}$ and $\mathbf{S} = \mathbf{U}\begin{bmatrix} 1 & 0 \\ 0 & \sigma_{\min} \end{bmatrix}\mathbf{V}$ be an SVD decomposition. Define $\mathbf{R} = \mathbf{U}\mathbf{V}^T$ and $\mathbf{Q} = \mathbf{U}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\mathbf{V}^T$. If $\det(\mathbf{R}) = -1$, swap $\mathbf{R}$ and $\mathbf{Q}$ and define $\alpha = \frac{1-\sigma_{\min}}{2}$, otherwise maintain $\mathbf{R}$ and $\mathbf{Q}$ and define $\alpha = \frac{\sigma_{\min}+1}{2}$. Then $\mathbf{S} = \alpha\mathbf{R} + (1-\alpha)\mathbf{Q}$ and $\alpha \in [0,1]$, $\mathbf{R} \in \mathsf{SO}(2)$, $\mathbf{Q} \in \mathsf{O}(2) - \mathsf{SO}(2)$ as desired.

- $\mathcal{C} \subset \mathbb{SS}$:
  Starting from $\alpha \in [0,1]$, $\mathbf{R} \in \mathsf{SO}(2)$ and $\mathbf{Q} \in \mathsf{O}(2) - \mathsf{SO}(2)$, the first step is to find matrices $\mathbf{U}$ and $\mathbf{V}$ in $\mathsf{O}(2)$ that satisfy $\mathbf{U}\mathbf{V}^T = \mathbf{R}$ and $\mathbf{U}\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}\mathbf{V}^T = \mathbf{Q}$. To do so, notice that $\mathbf{R}\mathbf{Q}^T =$

$\mathbf{U}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\mathbf{U}^T$, hence matrix $\mathbf{U}$ contains the eigenvectors of the matrix $\mathbf{R}\mathbf{Q}^T$. This is due to the fact that this matrix is in $\mathsf{O}(2)-\mathsf{SO}(2)$ which means it's symmetric. Similarly, notice as well that $\mathbf{R}^T\mathbf{Q}=\mathbf{V}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\mathbf{V}^T$, which again means that $\mathbf{V}$ contains the eigenvectors of this matrix.

Then define the matrix

$$\begin{aligned}\mathbf{S} &= \alpha\mathbf{R}+(1-\alpha)\mathbf{Q}\\ &= \alpha\mathbf{U}\mathbf{V}^T+(1-\alpha)\mathbf{U}\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\mathbf{V}^T\\ &= \mathbf{U}\begin{bmatrix}1 & 0\\ 0 & 2\alpha-1\end{bmatrix}\mathbf{V}^T\end{aligned}$$

which is an SVD decomposition of a sub-Stiefel matrix if $\alpha\geq 1/2$. If $\alpha<1/2$, define matrix $\mathbf{E}=\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ (notice it is an involution) and insert it above as

$$\begin{aligned}\mathbf{S} &= \mathbf{U}\begin{bmatrix}1 & 0\\ 0 & 2\alpha-1\end{bmatrix}\mathbf{E}^{-1}\mathbf{E}\mathbf{V}^T\\ &= \mathbf{U}\begin{bmatrix}1 & 0\\ 0 & 1-2\alpha\end{bmatrix}\underbrace{\mathbf{E}\mathbf{V}^T}_{\hat{\mathbf{V}}^T}\end{aligned}$$

which is an SVD decomposition of a sub-Stiefel matrix.

This completes the proof. ∎

This is an interesting result since it characterizes the sub-Stiefel matrices as the union of all line segments connecting $\mathsf{SO}(2)$ to $\mathsf{O}(2)-\mathsf{SO}(2)$, when seen as a subset of $\mathbb{R}^{2\times 2}$. It is also a nice representation for optimization since all sets are connected. In the SVD representation the orthogonal matrices could either be in $\mathsf{SO}(2)$ or $\mathsf{O}(2)-\mathsf{SO}(2)$ which means a jump from one set to the other had to be considered. Alternatively the SVD representation could be changed to consider only matrices in $\mathsf{SO}(2)$ as long as the minimum singular value would be allowed to be negative.

Topologically, the sub-Stiefel set is homeomorphic to a 3-dimensional sphere. This result is important since it provides intuition for its shape, and completely characterizes all topological properties

since they are inherited from the sphere, a well known topological manifold. For example, the next theorem hides the statement that the set $\mathbb{SS}$ is connected, even though this same property could have been easily proven otherwise.

Theorem A.10 *The set $\mathbb{SS}$ is homeomorphic to $\mathbb{S}(3)$.*

Proof Consider the function

$$\begin{aligned} f : \mathbb{SS} &\longrightarrow \mathbb{S}(3) \\ \mathbf{A} &\longmapsto \frac{\mathrm{vec}(\mathbf{A})}{\|\mathbf{A}\|} \end{aligned}$$

with corresponding inverse function

$$\begin{aligned} f^{-1} : \mathbb{S}(3) &\longrightarrow \mathbb{SS} \\ \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} &\longmapsto \frac{\begin{bmatrix} a & c \\ b & d \end{bmatrix}}{\sigma_{\max}\left(\begin{bmatrix} a & c \\ b & d \end{bmatrix}\right)} \end{aligned}$$

The function and its proposed inverse are continuous in open subsets of the ambient spaces containing the domain and the image, implying that both functions are continuous in the subspace topology ([21] section 18, page 108).

Let $\mathbf{S} \in \mathbb{SS}$ and $\mathbf{S} = \mathbf{U}\begin{bmatrix} 1 & 0 \\ 0 & \sigma_{\min} \end{bmatrix}\mathbf{V}^T$ an SVD decomposition. Then it is easily verified that the functions are inverse of each other since, apart from the reshape, one function scales the singular values to have unit norm, and the other rescales the singular values back to $\sigma_{\max} = 1$, both using positive scale factors and continuous functions.

Hence f is a homeomorphism, proving the theorem. ∎

Given a sub-Stiefel matrix, there are (in general) four ways to complete it to a 3×3 rotation matrix, of which two have positive determinant, and the other two have negative determinant. In the context of isometric reconstructions the negative determinant solutions are meaningless and one can restrict to the special orthogonal matrices instead of the whole orthogonal group. Unfortunately the the two remaining choices are ambiguous and there's no other physical characteristic to narrow the problem to a single solution. Only if the sub-Stiefel matrix is itself a rotation matrix does the ambiguity disappear,

and only a single completion is possible. Hence, if $\mathbf{S} = \begin{bmatrix} s_1^1 & s_2^1 \\ s_1^2 & s_2^2 \end{bmatrix}$ is a sub-Stiefel matrix it can be completed as

$$\underbrace{\begin{bmatrix} s_1^1 & s_2^1 \\ s_1^2 & s_2^2 \end{bmatrix}}_{\text{Sub-Stiefel Matrix}} \Rightarrow \underbrace{\begin{bmatrix} s_1^1 & s_2^1 & s_3^1 \\ s_1^2 & s_2^2 & s_3^2 \\ s_1^3 & s_2^3 & s_3^3 \end{bmatrix}}_{\text{Rotation Matrix 1}} \text{ or } \underbrace{\begin{bmatrix} s_1^1 & s_2^1 & -s_3^1 \\ s_1^2 & s_2^2 & -s_3^2 \\ -s_1^3 & -s_2^3 & s_3^3 \end{bmatrix}}_{\text{Rotation Matrix 2}}$$

by choosing a sign for $s_1^3 = \pm\sqrt{1-(s_1^1)^2-(s_1^2)^2}$ (if it is non-zero) then there a single choice for s_2^3 such that $(s_2^3)^2 = 1-(s_2^1)^2-s_2^(2)^2$ and the second column is orthogonal to the first. The third column is simply the cross product of the first two columns so as to yield a rotation matrix with positive determinant. As a curiosity note that $s_3^3 = \det(\mathbf{S})$ which is proved by inspection when writing the third coordinate of the cross product.

This section ends with a formula to compute the singular values of a 2×2 matrix in closed form.

Theorem A.11 *The singular values of a matrix $\mathbf{A} \in \mathsf{M}^{2\times 2}$ are given by the positive roots of the polynomial*

$$s^4 - \|\mathbf{A}\|^2 s^2 + \det(\mathbf{A})^2 = 0$$

Proof The roots of a second degree polynomial may be found by the quadratic equation. Writing the norm and determinant of $\mathbf{A}$ in terms of its singular values results in

$$\begin{aligned} s^2 &= \frac{\sigma_{\max}^2 + \sigma_{\min}^2 \pm \sqrt{\left(\sigma_{\max}^2 + \sigma_{\min}^2\right)^2 - 4\sigma_{\max}^2\sigma_{\min}^2}}{2} \\ &= \frac{\sigma_{\max}^2 + \sigma_{\min}^2 \pm \sqrt{\left(\sigma_{\max}^2 - \sigma_{\min}^2\right)^2}}{2} \end{aligned}$$

since the inside of the square root is always positive ($\sigma_{\max} \geq \sigma_{\min}$),

$$\begin{aligned} s^2 &= \frac{\sigma_{\max}^2 + \sigma_{\min}^2 \pm \left(\sigma_{\max}^2 - \sigma_{\min}^2\right)}{2} \\ &= \left\{\sigma_{\max}^2, \sigma_{\min}^2\right\} \end{aligned}$$

hence taking square roots yields the result. ∎

This ends the characterization of the sub-Stiefel set.

Appendix B

Sub-Stiefel Centering Problem

Solving the optimization problem defined in equation (3.7), here repeated for convenience

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i,k} \log^2 \left(\sigma_{\max}\left(\mathbf{M}_i^k\, \mathbf{H}/s^k\right)\right) \\ \text{s.t.} \quad & \mathbf{H} \in \mathbb{GL}(2) \\ & s^k \in \mathbb{R}^+ \end{aligned} \tag{B.1}$$

which is a non-convex and non-compact optimization problem, is not simple but a few tricks can be exploited. First, the $\sigma_{\max}$ function is invariant to orthogonal matrix multiplication on the right (i.e. $\sigma_{\max}\left(\mathbf{M}_i^k\, \mathbf{H}\right) = \sigma_{\max}\left(\mathbf{M}_i^k\, \mathbf{H}\, \mathbf{O}\right)$ for any $\mathbf{O} \in \mathsf{O}(2)$), meaning that these matrices can be factored out through a polar or QR decomposition on $\mathbf{H}$. This means that entry h_2^2 of matrix $\mathbf{H} = \left[h_j^i\right]$ can be forced to 0, reducing the number of variables to 3.

To simplify the discussion, first the special case of $s^k = 1$ is considered (which results in considering orthographic cameras instead of scaled orthographic cameras) and later extended to the general case.

B.1 Orthographic Cameras

The next simplification is less obvious, but will result in a compact 2 dimensional optimization problem. The key is to go to projective space by quotienting out the scale factor, using the fact that a scale factor commutes (as an absolute value) with the $\sigma_{\max}$ function. The optimization problem can

be written as

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i,k} \log^2 \left(|\lambda| \sigma_{\max} \left(\mathbf{M}_i^k \, \bar{\mathbf{H}} \right) \right) \\ \text{s.t.} \quad & \bar{\mathbf{H}} \in \mathbb{RP}^2 \\ & |\lambda| \in \mathbb{R}^+ \end{aligned}$$

Next rewrite the problem as follows

$$\begin{aligned} \text{minimize} \quad & \left(\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i,k} \log^2 \left(|\lambda| \sigma_{\max} \left(\mathbf{M}_i^k \, \bar{\mathbf{H}} \right) \right) \\ \text{s.t.} \quad & |\lambda| \in \mathbb{R}^+ \end{aligned} \right) \\ \text{s.t.} \quad & \bar{\mathbf{H}} \in \mathbb{RP}^2 \end{aligned}$$

in which the inner optimization problem is exactly equal to the optimization problem in equation (3.6) where the solution was already shown to be $\log(|\lambda^*|) = -\mathbb{E}\left[\log\left(\sigma_{\max}\left(\mathbf{M}_i^k \, \bar{\mathbf{H}}\right)\right)\right]$. This is exactly the property that made this choice of a distance function better than the traditional square of differences. The result can be plugged back resulting in

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_i \left(\log\left(\sigma_{\max}\left(\mathbf{M}_i^k \, \bar{\mathbf{H}}\right)\right) - \mathbb{E}\left[\log\left(\sigma_{\max}\left(\mathbf{M}_i^k \, \bar{\mathbf{H}}\right)\right)\right] \right)^2 \\ \text{s.t.} \quad & \bar{\mathbf{H}} \in \mathbb{RP}^2 \end{aligned}$$

Dividing the cost function by the number of $\mathbf{M}_i^k$ matrices, the minimizer is not changed and the cost function becomes a variance:

$$\begin{aligned} \text{minimize} \quad & \mathbb{VAR}_{i,k}\left[\log\left(\sigma_{\max}\left(\mathbf{M}_i^k \, \bar{\mathbf{H}}\right)\right)\right] \\ \text{s.t.} \quad & \bar{\mathbf{H}} \in \mathbb{RP}^2 \end{aligned}$$

The fact that it is written as a variance is just syntactic sugar at this point, but it does give intuition as to what is being done. Later, this will be exploited to impose different camera models but for now just think that reducing the problem domain to $\mathbb{RP}^2$ makes it significantly easier to solve, since it is a compact two dimensional differentiable manifold, definitely within the reach of branch and bound algorithms if nothing better is possible. Maybe equally important is that the function can now be visualized, allowing intuition and a clearer idea of how hard it is to solve (see figure B.1).

Unfortunately, as the figure shows, the problem is not convex. Worse is the fact that it sometimes has multiple local minima and that in the presence of significant noise it might even converge to a non

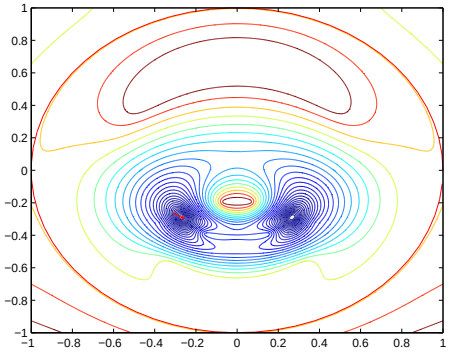


Figure B.1: Example level set of the sub-Stiefel centering problem cost function. A stereographic projection of $\mathbb{RP}^2$ was used as coordinates. Here blue lines represent low values, red lines represent high values.

invertible $\mathbf{H}$. Despite these shortcomings, when used to solve the problem at hand it does produce seemingly good results without much concern over which local minimum is used (usually the minima are very close together). Nonetheless, as future work, the effects of choosing a non-global local minimum should be further studied.

Since the $\sigma_{\max}$ is smooth almost everywhere (it is non-smooth when both eigenvalues are equal) and gradient vector and Hessian are computable for every function involved (see [24] for details on how to compute Hessians and gradients of the maximum singular value), it is relatively straightforward to implement a Newton-like method on the projective space.

Once a solution $\bar{\mathbf{H}}^*$ has been found, λ^* follows naturally since it has the closed form expression shown above. Note that this provides a solution up to a global rotation (factored out in the QR decomposition above). This is to be expected since no global coordinate system has been imposed.

B.2 Scale Orthographic Camera Models

Now that the solution for orthographic cameras has been revealed, a simple trick is used to solve the case of scaled orthographic cameras. The optimization problem that needs to be solved is

$$\begin{aligned} \text{minimize} \quad & \textstyle\sum_{i,k} d^2_{\mathbb{SS}} \left(\mathbf{M}_i^{k*} \, \mathbf{H} / s^k \right) \\ \text{s.t.} \quad & \mathbf{H} \in \mathbb{GL}(2) \\ & s^k \in \mathbb{R}^+ \end{aligned}$$

Here using the chosen cost function $d_{\mathbb{SS}}(\mathbf{X}) = |\log(\sigma_{\max}(\mathbf{X}))|$. Interestingly, the same trick used in the previous discussion to go to projective space will be re-used to allow use of the slightly more complicated camera model. Again, the optimization problem is separated as

$$\begin{array}{rl} \text{minimize} & \sum_i \left(\begin{array}{rl} \text{minimize} & \sum_k d^2_{\mathbb{SS}} \left(\mathbf{M}_i^{k*} \, \mathbf{H} / s^k \right) \\ \text{s.t.} & s^k \in \mathbb{R}^+ \end{array} \right) \\ \text{s.t.} & \mathbf{H} \in \mathbb{GL}(2) \end{array}$$

and remembering the discussion of using λ to write a variance cost function in section B.1, the same trick is used using each s^k, separately, instead. This becomes

$$\begin{array}{rl} \text{minimize} & \sum_i \mathbb{VAR}_k \left[\log \left(\sigma_{\max} \left(\mathbf{M}_i^{k*} \, \mathbf{H} \right) \right) \right] \\ \text{s.t.} & \mathbf{H} \in \mathbb{GL}(2) \end{array}$$

Notice that now the scale factor λ has no effect since it propagates outside the $\log$ additively which has no impact on the variance. This means that there's an additional ambiguity in the final reconstruction (adding to the global rotation matrix factored in the previous QR decomposition) which is expected since the most that can be hoped for is a reconstruction that leaves the camera model invariant. Thus the final optimization problem is

$$\begin{array}{rl} \text{minimize} & \sum_i \mathbb{VAR}_k \left[\log \left(\sigma_{\max} \left(\mathbf{M}_i^{k*} \, \bar{\mathbf{H}} \right) \right) \right] \\ \text{s.t.} & \bar{\mathbf{H}} \in \mathbb{RP}^2 \end{array}$$

Which is a sum of problems similar to the one that was solved in the previous section, easily extended in the Newton method. Once a solution for $\bar{\mathbf{H}}$ is found, the solution for the scale factors s^k is simply $\log(s^{k*}) = -\mathbb{E}_k \left[\log \left(\sigma_{\max} \left(\mathbf{M}_i^{k*} \, \mathbf{H}^* \right) \right) \right]$, where any scale factor can be used to go from $\bar{\mathbf{H}} \in \mathbb{RP}^2$ to $\mathbf{H} \in \mathbb{GL}(2)$. The scale ambiguity is present since if $\mathbf{H}$ is scaled, so will the s^{k*}.

Appendix C

Sub-Stiefel Procrustes Problem

In matrix approximation theory, there's an important class of problems which try to estimate the best linear transformation satisfying certain properties that best describe the observed data. The general problem is written as

$$\begin{aligned} \text{minimize} \quad & \left\|\mathbf{V} - \mathbf{X}\mathbf{W}\right\|^2 \\ \text{s.t.} \quad & \mathbf{X} \in \{\text{matrices with a certain property}\} \end{aligned}$$

Under this class of problems, when no particular property is desired for the matrix $\mathbf{X}$ the problem is known as least squares fitting. When the number of rows of the matrices $\mathbf{V}$ and $\mathbf{W}$ is the same and $\mathbf{X}$ is constrained to be orthogonal ($\mathbf{X}\mathbf{X}^T = \mathbf{I}$) the problem is known as the Orthogonal Procrustes Problem and has a simple solution involving a singular value decomposition. In this section a new problem in this class, the **sub-Stiefel Procrustes problem**, is proposed where $\mathbf{X} \in \mathbb{SS}$:

Problem Statement C.1 (Sub-Stiefel Procrustes Problem) *Solve*

$$\begin{aligned} \textit{minimize} \quad & \left\|\mathbf{V} - \mathbf{X}\mathbf{W}\right\|^2 \\ \textit{s.t.} \quad & \mathbf{X} \in \mathbb{SS} \end{aligned} \tag{C.1}$$

Although computationally involved, the solution of this problem is exact up to finding the real roots of a 6 degree polynomial. The solution involves the use of Gröbner basis from Algebraic Geometry to solve a system of polynomial equations, but note that this time consuming step only needs to be done once (here) and the final algorithm takes very little time to run being practically instantaneous on modern hardware. In the end the algorithm does not need to compute a Gröbner basis with each run.

Using theorem A.5 the problem in equation (C.1) can be rewritten as

$$\begin{aligned} \text{minimize} \quad & \left\|\mathbf{V}-\mathbf{OW}\right\|^2 \\ \text{s.t.} \quad & \sigma_{\max}(\mathbf{O}) = 1 \\ & \mathbf{O} \in \mathsf{M}^{2\times 2} \end{aligned}$$

Using γ as a Lagrangian multiplier, the Lagrangian function is then

$$\mathcal{L} = \operatorname{tr}\left\{(\mathbf{V}-\mathbf{OW})^T(\mathbf{V}-\mathbf{OW})\right\} + \gamma\left(\sigma_{\max}(\mathbf{O})-1\right)$$

the function $\sigma_{\max}(\cdot)$ is differentiable everywhere except when both eigenvalues are equal (see [24]). This case is handled separately later. If $\mathbf{O} = \mathbf{S}\begin{bmatrix}\sigma_{\max} & 0\\ 0 & \sigma_{\min}\end{bmatrix}\mathbf{T}^T$ is a singular value decomposition then

$$\frac{\partial\sigma_{\max}(\mathbf{O})}{\partial\mathbf{O}} = \mathbf{s}_{\max}{\mathbf{t}_{\max}}^T$$

where $\mathbf{s}_{\max}$ and $\mathbf{t}_{\max}$ are the columns of $\mathbf{S}$ and $\mathbf{T}$ corresponding to the maximum singular value.

Hence, the Karush-Kuhn-Tucker system for this problem is (see [19] for matrix derivative rules and a constrained optimization book such as [22] for the Karush-Kuhn-Tucker conditions):

$$\begin{aligned} -\mathbf{VW}^T + \mathbf{OWW}^T + \gamma{\mathbf{s}_{\max}}{\mathbf{t}_{\max}}^T &= 0 \\ \sigma_{\max}(\mathbf{O}) &= 1 \end{aligned}$$

the last constraint can be implicitly included in the first using the previously mentioned SVD decomposition:

$$\begin{aligned} -\mathbf{VW}^T + \mathbf{S}\begin{bmatrix}1 & 0\\ 0 & \sigma_{\min}\end{bmatrix}\mathbf{T}^T\mathbf{WW}^T + \mathbf{S}\begin{bmatrix}\gamma & 0\\ 0 & 0\end{bmatrix}\mathbf{T}^T &= 0 \\ \mathbf{S}^T\mathbf{S} &= \mathbf{I} \\ \mathbf{T}^T\mathbf{T} &= \mathbf{I} \\ \sigma_{\min} &\leq 1 \end{aligned} \tag{C.2}$$

There's a hidden subtlety with the current problem formulation. The constraint $\sigma_{\min} \leq 1$ involves enumerating both singular values and choosing the lowest. It also hides the afore-mentioned

problem of non-differentiability when the lowest singular value is equal to 1. This is not trivial in an optimization setting, hence the problem is going to be split into two separate optimization problems.

First, notice that if $x_i \in \mathcal{D}$ enumerates all the critical points of a given function with domain $\mathcal{D}$, then the critical points of the same function restricted to a closed (with respect to $\mathcal{D}$) subdomain $\mathcal{E} \subset \mathcal{D}$ are $\{x_i\} \bigcap \mathcal{E}$ joined with the critical points of the function restricted to the border of $\mathcal{E}$ in $\mathcal{D}$. In the problem at hand consider $\mathcal{D} = \{\text{Set of } 2 \times 2 \text{ matrices with at least one singular value equal to } 1\}$ and $\mathcal{E} = \mathbb{SS}$. The border in this case will be the 2×2 Orthogonal matrices where both singular values are equal to 1. Hence, if all critical points of the cost function for the relaxed problem can be found, then either the original problem's global minimum is in these points intersected with the domain or is given by a simple Procrustes problem.

Notice that if the last constraint is removed, the problem consists strictly of polynomial equalities, hence algebraic geometry is able to solve the system using Gröbner basis. Unfortunately the equations are still too complex to tackle with modern computer algebra systems (such as Maple and Mathematica) and any attempt made by the author to tackle the problem directly quickly exhausted the computational resources of a modest desktop computer. The problem with the equations (C.2) is that it involves too many different symbols which quickly choke the software. Fortunately there are a few tricks which can be used to reduce the complexity.

First pre and post multiply by $\mathbf{S}^T$ and $\mathbf{T}^T$ so that the first equation becomes

$$-\mathbf{S}^T\mathbf{V}\mathbf{W}^T\mathbf{T} + \begin{bmatrix} 1 & 0 \\ 0 & \sigma_{\min} \end{bmatrix} \mathbf{T}^T\mathbf{W}\mathbf{W}^T\mathbf{T} + \begin{bmatrix} \gamma & 0 \\ 0 & 0 \end{bmatrix} = 0 \tag{C.3}$$

Now pre and post multiply by the vectors $\begin{bmatrix} 1 & 0 \end{bmatrix}$ and $\begin{bmatrix} 0 & 1 \end{bmatrix}^T$ which generates a scalar equation without any dependencies on σ or γ:

$$-\mathbf{s}_1^T\mathbf{V}\mathbf{W}^T\mathbf{t}_2 + \mathbf{t}_1^T\mathbf{W}\mathbf{W}^T\mathbf{t}_2 = 0$$

Where the new vectors are the matrices' columns as $\mathbf{S} = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 \end{bmatrix}$ $\mathbf{T} = \begin{bmatrix} \mathbf{t}_1 & \mathbf{t}_2 \end{bmatrix}$.

Noting that $\mathsf{O}(2)$ is a one dimensional manifold, one needs at least a second equation for a solution to be obtained (since there are two orthogonal matrices). To obtain a second equation which only depends on $\mathbf{T}$ and $\mathbf{S}$, rewrite the equation as

$$\left(-\mathbf{S}^T\mathbf{V}\mathbf{W}^T\left(\mathbf{W}\mathbf{W}^T\right)^{-1}\mathbf{T} + \begin{bmatrix} 1 & 0 \\ 0 & \sigma_{\min} \end{bmatrix}\right) \mathbf{T}^T\mathbf{W}\mathbf{W}^T\mathbf{T} + \begin{bmatrix} \gamma & 0 \\ 0 & 0 \end{bmatrix} = 0 \tag{C.4}$$

Since the points are assumed to be non-degenerate, this can be re-written as

$$-\mathbf{S}^T\mathbf{V}\mathbf{W}^T\left(\mathbf{W}\mathbf{W}^T\right)^{-1}\mathbf{T}+\begin{bmatrix}1 & 0\\ 0 & \sigma_{\min}\end{bmatrix}+\begin{bmatrix}\gamma & 0\\ 0 & 0\end{bmatrix}\mathbf{T}^T\left(\mathbf{W}\mathbf{W}^T\right)^{-1}\mathbf{T}=0 \tag{C.5}$$

And now pre and post multiply by the vectors $\begin{bmatrix}0 & 1\end{bmatrix}$ and $\begin{bmatrix}1 & 0\end{bmatrix}^T$ which yields the second equation

$$\mathbf{s}_2^T\mathbf{V}\mathbf{W}^T\left(\mathbf{W}\mathbf{W}^T\right)^{-1}\mathbf{t}_1=0 \tag{C.6}$$

Since a matrix in $\mathsf{SO}(2)$ can be parameterized as $\begin{bmatrix}c & s\\ -s & c\end{bmatrix}$ where s and c obey the constraint $s^2+c^2=1$, the polynomial system of equations is reduced to 4 variables and 4 equations. A question arises on whether or not the orthogonal matrices of the SVD decomposition can be assumed to be in $\mathsf{SO}(2)$. The answer is yes, as long as $\sigma_{\min}$ is allowed to take negative values.

Before writing the explicit system lets further reduce the number of constants that appear in the description. To do so, notice that a change of variables in the optimization problem in statement C.1 can be used to impose that $\mathbf{W}\mathbf{W}^T$ is diagonal and the matrix $\mathbf{V}\mathbf{W}^T$ is upper triangular. To do so, suppose that $\mathbf{E}$ is the orthogonal matrix of an eigenvalue decomposition of the symmetric matrix $\mathbf{W}\mathbf{W}^T$ and that $\mathbf{Q}$ is the orthogonal matrix of a QR decomposition of $\mathbf{V}\mathbf{W}^T\mathbf{E}$. Since the norm is left and right invariant to multiplication by orthogonal matrices,

$$\left\|\mathbf{V}-\mathbf{X}\mathbf{W}\right\|^2=\|\underbrace{\mathbf{R}\mathbf{V}}_{\hat{\mathbf{V}}}-\underbrace{\mathbf{R}^T\mathbf{X}\mathbf{E}}_{\hat{\mathbf{X}}}\underbrace{\mathbf{E}^T\mathbf{W}}_{\hat{\mathbf{W}}}\|^2$$

Theorem A.1 guarantees that $\hat{\mathbf{X}}$ is sub-Stieffel and by construction $\hat{\mathbf{W}}\hat{\mathbf{W}}^T$ is diagonal and the matrix $\hat{\mathbf{V}}\hat{\mathbf{W}}^T$ is upper triangular.

There's a further redundancy in the cost function due to the fact that the minimizer does not change if a scale factor is applied. thus the cost function can be divided by a constant so that the first entry of $\mathbf{W}\mathbf{W}^T$ is 1.

If $\mathbf{V}\mathbf{W}^T\left(\mathbf{W}\mathbf{W}^T\right)^{-1} = \begin{bmatrix} x_1 & x_2 \\ 0 & x_4 \end{bmatrix}$ and $\mathbf{W}\mathbf{W}^T = \begin{bmatrix} 1 & 0 \\ 0 & b \end{bmatrix}$ then the following system can be written explicitly

$$\begin{aligned} s_1^2 + s_2^2 - 1 &= 0 \\ t_1^2 + t_2^2 - 1 &= 0 \\ x_1 t_2 s_1 - b x_2 s_1 t_1 + b x_4 t_1 s_2 + b t_2 t_1 - t_1 t_2 &= 0 \\ -x_1 s_2 t_1 - x_2 s_2 t_2 + x_4 s_1 t_2 &= 0 \end{aligned}$$

where the orthogonal matrices, which are the variables of the problem, were written as $\mathbf{S} = \begin{bmatrix} s_1 & -s_2 \\ s_2 & s_1 \end{bmatrix}$ and $\mathbf{T} = \begin{bmatrix} t_1 & -t_2 \\ t_2 & t_1 \end{bmatrix}$. This system is finally solvable with Maple. Without going into too much detail on algebraic geometry (a very good book on the subject is [6]) the system has 12 solutions which some of which might be complex. Using Gröbner basis techniques, this polynomial system (or, in the language of the field, this "ideal") can be re-written as another system where the second equation depends on less variables, than the first, the third depends on less variables than the second, etc, up to the last equation only depending on a single variable. This is analogous to the QR matrix decomposition in linear algebra, which is in fact a special case.

Although the actual coefficients of the polynomial (in the variable t_1) are shown later, for now it's enough to state that only the even coefficients are non-zero and thus it can be written as

$$a_{12} t_1^{12} + a_{10} t_1^{10} + a_8 t_1^8 + a_6 t_1^6 + a_4 t_1^4 + a_2 t_1^2 + a_0 = 0$$

where the coefficients a_i depend only on the constants of the problem (x_1, x_2, x_4 and b). Since the odd powers are all zero, the zeros of this polynomial may be obtained from a polynomial of degree 6, where each root counts twice with a different sign. Of these, only the real (since numeric errors are unavoidable, one should check the ones that are close to being real as well) are of interest.

From here, one can consider the second polynomial given by the Gröbner basis, but since its expression is very ugly a more direct approach is attempted. For each of the possible solutions for t_1, two possible solutions are possible for t_2 since they form a column of an orthogonal matrix. Hence $t_2 = \pm\sqrt{1 - t_1^2}$ and two possibilities for matrix $\mathbf{T}$ are obtained. These possibilities must be kept and later checked for consistency.

Next, equation (C.5) holds implicitly the information that the matrix $\mathbf{S}$ is the orthogonal factor of a QR decomposition of $\mathbf{V}\mathbf{W}^T\left(\mathbf{W}\mathbf{W}^T\right)^{-1}\mathbf{T}$. This is due to (almost) uniqueness of the decomposition and the fact that every other term in the equation has the lower left entry equal to zero. Also, in the same QR decomposition, the non-zero entry of the second line of the triangular matrix must be equal to $\sigma_{\min}$. Depending on the implementation of the QR decomposition, if the resulting $\sigma_{\min}$ value is negative, change its sign and the sign of the second row of $\mathbf{S}$. Finally, both signs of the first row of $\mathbf{S}$ are possible, hence for each possible matrix $\mathbf{T}$, two possibilities for $\mathbf{S}$ are obtained. Since the variable γ is of no interest, there's no point in computing it.

After all the possible solutions are enumerated, they must be checked and any solution which does not obey equation (C.3) or if $\sigma_{\min} > 1$, must be discarded. As discussed previously, the solutions which are left must be compared with the solution on the border of the set, which is obtained from a simple orthogonal Procrustes problem (see for example chapter 4 of [12]). All these solutions are evaluated in the cost function and the one that produces the least cost is chosen.

Finally, in the previous discussion the polynomial coefficients of the Gröbner basis were not written explicitly. They are listed in table C.1 for reference purposes.

$$
\begin{aligned}
v_{12} &= (b-1)^4\,(x_1^2 - 2\,x_1\,x_4 + x_2^2 + x_4^2)\,(x_1^2 + 2\,x_1\,x_4 + x_2^2 + x_4^2)\\
v_{10} &= 2\,(b-1)^2\,(b^2\,x_1^4\,x_2^2 - b^2\,x_1^4 + 2\,b^2\,x_1^2\,x_2^4 + b^2\,x_1^2\,x_2^2\,x_4^2 - 3\,b^2\,x_1^2\,x_2^2 - b^2\,x_1^2\,x_4^4 + 3\,b^2\,x_1^2\,x_4^2 + b^2\,x_2^6 + 3\,b^2\,x_2^4\,x_4^2 - 2\,b^2\,x_2^4\\
&\quad + 3\,b^2\,x_2^2\,x_4^4 - 4\,b^2\,x_2^2\,x_4^2 + b^2\,x_4^6 - 2\,b^2\,x_4^4 + 2\,b\,x_1^4\,x_4^2 + 2\,b\,x_1^4 - 2\,b\,x_1^2\,x_2^2\,x_4^2 + 6\,b\,x_1^2\,x_2^2 - 2\,b\,x_1^2\,x_4^4 - 6\,b\,x_1^2\,x_4^2 + 4\,b\,x_2^4\\
&\quad + 8\,b\,x_2^2\,x_4^2 + 4\,b\,x_4^4 - x_1^6 - 2\,x_1^4\,x_2^2 + x_1^4\,x_4^2 - x_1^4 - x_1^2\,x_2^4 - x_1^2\,x_2^2\,x_4^2 - 3\,x_1^2\,x_2^2 + 3\,x_1^2\,x_4^2 - 2\,x_2^4 - 4\,x_2^2\,x_4^2 - 2\,x_4^4)\\
v_8 &= (b^4\,x_1^4\,x_2^4 - 2\,b^4\,x_1^4\,x_2^2 + b^4\,x_1^4 + 2\,b^4\,x_1^2\,x_2^6 + 4\,b^4\,x_1^2\,x_2^4\,x_4^2 - 8\,b^4\,x_1^2\,x_2^4 + 2\,b^4\,x_1^2\,x_2^2\,x_4^4 - 4\,b^4\,x_1^2\,x_2^2\,x_4^2 + 6\,b^4\,x_1^2\,x_2^2\\
&\quad + 4\,b^4\,x_1^2\,x_4^4 - 6\,b^4\,x_1^2\,x_4^2 + b^4\,x_2^8 + 4\,b^4\,x_2^6\,x_4^2 - 6\,b^4\,x_2^6 + 6\,b^4\,x_2^4\,x_4^4 - 18\,b^4\,x_2^4\,x_4^2 + 6\,b^4\,x_2^4 + 4\,b^4\,x_2^2\,x_4^6 - 18\,b^4\,x_2^2\,x_4^4 + 12\,b^4\,x_2^2\,x_4^2\\
&\quad + b^4\,x_4^8 - 6\,b^4\,x_4^6 + 6\,b^4\,x_4^4 - 4\,b^3\,x_1^4\,x_2^2\,x_4^2 + 4\,b^3\,x_1^4\,x_2^2 - 8\,b^3\,x_1^4\,x_4^2 - 4\,b^3\,x_1^4 - 4\,b^3\,x_1^2\,x_2^4\,x_4^2 + 16\,b^3\,x_1^2\,x_4^2 - 8\,b^3\,x_1^2\,x_2^2\,x_4^4\\
&\quad + 20\,b^3\,x_1^2\,x_2^2\,x_4^2 - 24\,b^3\,x_1^2\,x_2^2 - 4\,b^3\,x_1^2\,x_4^6 + 4\,b^3\,x_1^2\,x_4^4 + 24\,b^3\,x_1^2\,x_4^2 + 12\,b^3\,x_2^6 + 36\,b^3\,x_2^4\,x_4^2 - 24\,b^3\,x_2^4 + 36\,b^3\,x_2^2\,x_4^4 - 48\,b^3\,x_2^2\,x_4^2\\
&\quad + 12\,b^3\,x_4^6 - 24\,b^3\,x_4^4 + 2\,b^2\,x_1^6\,x_2^2 + 4\,b^2\,x_1^6 + 4\,b^2\,x_1^4\,x_2^4 + 4\,b^2\,x_1^4\,x_2^2\,x_4^2 + 10\,b^2\,x_1^4\,x_2^2 + 6\,b^2\,x_1^4\,x_4^4 + 10\,b^2\,x_1^4\,x_4^2 + 6\,b^2\,x_1^4\\
&\quad + 2\,b^2\,x_1^2\,x_2^6 + 4\,b^2\,x_1^2\,x_2^4\,x_4^2 + 2\,b^2\,x_1^2\,x_2^2\,x_4^4 - 20\,b^2\,x_1^2\,x_2^2\,x_4^2 + 36\,b^2\,x_1^2\,x_2^2 - 20\,b^2\,x_1^2\,x_4^4 - 36\,b^2\,x_1^2\,x_4^2 - 6\,b^2\,x_2^6 - 18\,b^2\,x_2^4\,x_4^2\\
&\quad + 36\,b^2\,x_2^4 - 18\,b^2\,x_2^2\,x_4^4 + 72\,b^2\,x_2^2\,x_4^2 - 6\,b^2\,x_4^6 + 36\,b^2\,x_4^4 - 4\,b\,x_1^6\,x_2^2 - 8\,b\,x_1^6 - 4\,b\,x_1^4\,x_2^2\,x_4^2 - 24\,b\,x_1^4\,x_2^2 + 4\,b\,x_1^4\,x_4^2 - 4\,b\,x_1^4\\
&\quad - 16\,b\,x_1^2\,x_2^4 - 4\,b\,x_1^2\,x_2^2\,x_4^2 - 24\,b\,x_1^2\,x_2^2 + 12\,b\,x_1^2\,x_4^4 + 24\,b\,x_1^2\,x_4^2 - 24\,b\,x_2^2 - 48\,b\,x_2^2\,x_4^2 - 24\,b\,x_4^4 + x_1^8 + 2\,x_1^6\,x_2^2 + 4\,x_1^6 + x_1^4\,x_2^4\\
&\quad + 12\,x_1^4\,x_2^2 - 6\,x_1^4\,x_4^2 + x_1^4 + 8\,x_1^2\,x_2^4 + 8\,x_1^2\,x_2^2\,x_4^2 + 6\,x_1^2\,x_2^2 - 6\,x_1^2\,x_4^2 + 6\,x_2^4 + 12\,x_2^2\,x_4^2 + 6\,x_4^4)\\
v_6 &= -(2\,b^4\,x_1^2\,x_2^6 + 4\,b^4\,x_1^2\,x_2^4\,x_4^2 - 4\,b^4\,x_1^2\,x_2^4 + 2\,b^4\,x_1^2\,x_2^2\,x_4^4 - 2\,b^4\,x_1^2\,x_2^2\,x_4^2 + 2\,b^4\,x_1^2\,x_2^2 + 2\,b^4\,x_1^2\,x_4^4 - 2\,b^4\,x_1^2\,x_4^2 + 2\,b^4\,x_2^8\\
&\quad + 8\,b^4\,x_2^6\,x_4^2 - 6\,b^4\,x_2^6 + 12\,b^4\,x_2^4\,x_4^4 - 18\,b^4\,x_2^4\,x_4^2 + 4\,b^4\,x_2^4 + 8\,b^4\,x_2^2\,x_4^6 - 18\,b^4\,x_2^2\,x_4^4 + 8\,b^4\,x_2^2\,x_4^2 + 2\,b^4\,x_4^8 - 6\,b^4\,x_4^6 + 4\,b^4\,x_4^4\\
&\quad - 4\,b^3\,x_1^4\,x_2^2\,x_4^2 - 4\,b^3\,x_1^4\,x_4^2 - 8\,b^3\,x_1^2\,x_2^4\,x_4^2 + 8\,b^3\,x_1^2\,x_4^4 - 16\,b^3\,x_1^2\,x_2^2\,x_4^4 + 16\,b^3\,x_1^2\,x_2^2\,x_4^2 - 8\,b^3\,x_1^2\,x_2^2 - 8\,b^3\,x_1^2\,x_4^6 + 8\,b^3\,x_1^2\,x_4^4\\
&\quad + 8\,b^3\,x_1^2\,x_4^2 + 12\,b^3\,x_2^6 + 36\,b^3\,x_2^4\,x_4^2 - 16\,b^3\,x_2^4 + 36\,b^3\,x_2^2\,x_4^4 - 32\,b^3\,x_2^2\,x_4^2 + 12\,b^3\,x_4^6 - 16\,b^3\,x_4^4 + 2\,b^2\,x_1^6\,x_2^2 + 2\,b^2\,x_1^6\\
&\quad + 8\,b^2\,x_1^4\,x_2^4 + 8\,b^2\,x_1^4\,x_2^2\,x_4^2 + 12\,b^2\,x_1^4\,x_2^2 + 12\,b^2\,x_1^4\,x_4^4 + 2\,b^2\,x_1^4\,x_4^2 + 6\,b^2\,x_1^2\,x_2^6 + 12\,b^2\,x_1^2\,x_2^4\,x_4^2 + 8\,b^2\,x_1^2\,x_2^4 + 6\,b^2\,x_1^2\,x_2^2\,x_4^4\\
&\quad - 14\,b^2\,x_1^2\,x_2^2\,x_4^2 + 12\,b^2\,x_1^2\,x_2^2 - 22\,b^2\,x_1^2\,x_4^4 - 12\,b^2\,x_1^2\,x_4^2 - 6\,b^2\,x_2^6 - 18\,b^2\,x_2^4\,x_4^2 + 24\,b^2\,x_2^4 - 18\,b^2\,x_2^2\,x_4^4 + 48\,b^2\,x_2^2\,x_4^2 - 6\,b^2\,x_4^6\\
&\quad + 24\,b^2\,x_4^4 - 8\,b\,x_1^6\,x_4^2 - 4\,b\,x_1^6 - 12\,b\,x_1^4\,x_2^2\,x_4^2 - 24\,b\,x_1^4\,x_2^2 + 8\,b\,x_1^4\,x_4^2 - 24\,b\,x_1^2\,x_4^4 - 12\,b\,x_1^2\,x_2^2\,x_4^2 - 8\,b\,x_1^2\,x_2^2 + 12\,b\,x_1^2\,x_4^4 + 8\,b\,x_1^2\,x_4^2\\
&\quad - 16\,b\,x_2^4 - 32\,b\,x_2^2\,x_4^2 - 16\,b\,x_4^4 + 2\,x_1^8 + 6\,x_1^6\,x_2^2 + 2\,x_1^6 + 4\,x_1^4\,x_2^4 + 12\,x_1^4\,x_2^2 - 6\,x_1^4\,x_4^2 + 12\,x_1^2\,x_2^4 + 12\,x_1^2\,x_2^2\,x_4^2 + 2\,x_1^2\,x_2^2 - 2\,x_1^2\,x_4^2\\
&\quad + 4\,x_2^4 + 8\,x_2^2\,x_4^2 + 4\,x_4^4)\\
v_4 &= (b^4\,x_2^8 + 4\,b^4\,x_2^6\,x_4^2 - 2\,b^4\,x_2^6 + 6\,b^4\,x_2^4\,x_4^4 - 6\,b^4\,x_2^4\,x_4^2 + b^4\,x_2^4 + 4\,b^4\,x_2^2\,x_4^6 - 6\,b^4\,x_2^2\,x_4^4 + 2\,b^4\,x_2^2\,x_4^2 + b^4\,x_4^8 - 2\,b^4\,x_4^6 + b^4\,x_4^4\\
&\quad - 4\,b^3\,x_1^2\,x_2^4\,x_4^2 - 8\,b^3\,x_1^2\,x_2^2\,x_4^4 + 4\,b^3\,x_1^2\,x_2^2\,x_4^2 - 4\,b^3\,x_1^2\,x_4^6 + 4\,b^3\,x_1^2\,x_4^4 + 4\,b^3\,x_2^6 + 12\,b^3\,x_2^4\,x_4^2 - 4\,b^3\,x_2^4 + 12\,b^3\,x_2^2\,x_4^4 - 8\,b^3\,x_2^2\,x_4^2\\
&\quad + 4\,b^3\,x_4^6 - 4\,b^3\,x_4^4 + 4\,b^2\,x_1^4\,x_2^4 + 4\,b^2\,x_1^4\,x_2^2\,x_4^2 + 4\,b^2\,x_1^4\,x_2^2 + 6\,b^2\,x_1^4\,x_4^4 - 2\,b^2\,x_1^4\,x_4^2 + 6\,b^2\,x_1^2\,x_2^6 + 12\,b^2\,x_1^2\,x_2^4\,x_4^2 + 8\,b^2\,x_1^2\,x_2^4\\
&\quad + 6\,b^2\,x_1^2\,x_2^2\,x_4^4 - 8\,b^2\,x_1^2\,x_4^4 - 2\,b^2\,x_2^6 - 6\,b^2\,x_2^4\,x_4^2 + 6\,b^2\,x_2^4 - 6\,b^2\,x_2^2\,x_4^4 + 12\,b^2\,x_2^2\,x_4^2 - 2\,b^2\,x_4^6 + 6\,b^2\,x_4^4 - 4\,b\,x_1^6\,x_4^2 - 12\,b\,x_1^4\,x_2^2\,x_4^2\\
&\quad - 8\,b\,x_1^4\,x_2^2 + 4\,b\,x_1^4\,x_4^2 - 16\,b\,x_1^2\,x_4^2 - 12\,b\,x_1^2\,x_2^2\,x_4^2 + 4\,b\,x_1^2\,x_4^4 - 4\,b\,x_2^4 - 8\,b\,x_2^2\,x_4^2 - 4\,b\,x_4^4 + x_1^8 + 6\,x_1^6\,x_2^2 + 6\,x_1^4\,x_2^4 + 4\,x_1^4\,x_2^2 - 2\,x_1^4\,x_4^2\\
&\quad + 8\,x_1^2\,x_2^4 + 8\,x_1^2\,x_2^2\,x_4^2 + x_2^4 + 2\,x_2^2\,x_4^2 + x_4^4)\\
v_2 &= -(2\,b^2\,x_1^2\,x_2^6 + 4\,b^2\,x_1^2\,x_2^4\,x_4^2 + 2\,b^2\,x_1^2\,x_2^4 + 2\,b^2\,x_1^2\,x_2^2\,x_4^4\\
&\quad + 2\,b^2\,x_1^2\,x_2^2\,x_4^2 - 4\,b\,x_1^4\,x_2^2\,x_4^2 - 4\,b\,x_1^2\,x_4^4 - 4\,b\,x_1^2\,x_2^2\,x_4^2 + 2\,x_1^6\,x_2^2 + 4\,x_1^4\,x_2^4 + 2\,x_1^4\,x_2^2 + 2\,x_1^2\,x_2^2\,x_4^2)\\
v_0 &= x_1^4\,x_2^4
\end{aligned}
$$

Table C.1: Sub-Stiefel Procrustes polynomial coefficients.

Bibliography

[1] Z. Bai, J. Demmel, J. Dongarra, A. Ruhe, and H. van der Vorst. *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. Society for Industrial and Applied Mathematics, 2000.

[2] Marcel Berger. *A Panoramic View of Riemannian Geometry*. Springer-Verlag, 2003.

[3] William M. Boothby. *An introduction to differentiable manifolds and Riemannian geometry*. Academic Press, New York :, 1975.

[4] H.-Y. Chen, I.-K. Lee, S. Leopoldseder, H. Pottmann, T. Randrup, and J. Wallner. On surface approximation using developable surfaces. *Graph. Models Image Process.*, 61(2):110–124, 1999.

[5] Pei Chen. Optimization algorithms on subspaces: Revisiting missing data problem in low-rank matrix. *Int. J. Comput. Vision*, 80(1):125–142, 2008.

[6] David A. Cox, John Little, and Donal O'Shea. *Ideals, Varieties, and Algorithms: An Introduction to Computational Algebraic Geometry and Commutative Algebra, 3/e (Undergraduate Texts in Mathematics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2007.

[7] Manfredo P. do Carmo. *Differential Geometry of Curves and Surfaces*. Prentice-Hall, Inc., Upper Saddle River, New Jersey, 1976.

[8] Ricardo Ferreira, João Xavier, and João Costeira. Reconstruction of isometrically deformable flat surfaces in 3d from multiple camera images. *ICASSP*, 2009.

[9] Ricardo Ferreira, João Xavier, and João Costeira. Shape from motion of nonrigid objects: The case of isometrically deformable flat surfaces. *British Machine Vision Conference - BMVC'09*, 2009.

[10] F.Perez and J.A. Suarez. Quasi-developable b-spline surfaces in ship hull design. *Computer Aided Design*, (39):853–862, 2007.

[11] Dmitry Fuchs and Serge Tabachnikov. More on paperfolding. *The American Mathematical Monthly*, 106(1):27–35, 1999.

[12] Dijksterhuis G.B. Gower J.C. *Procrustes Problems*. Oxford University Press, 2005.

[13] Rui F.C. Guerreiro and Pedro M.Q. Aguiar. Factorization with missing data for 3d structure recovery. *IEEE Workshop on Multimedia Signal Processing*, pages 105–108, 2002.

[14] Nail Gumerov, Ali Z, Ramani Duraiswami, and Larry S. Davis. Structure of applicable surfaces from single views. pages 482–496, 2004.

[15] Horn and C. R. Johnson. *Matrix Analysis*. Cambridge University Press, 1985.

[16] S. Kobayashi and K. Nomizu. *Foundations of Differential Geometry*. Wiley, New York, 1963.

[17] John M. Lee. *Riemannian Manifolds: An Introduction to Curvature*. Springer, 1997.

[18] S. Leopoldseder and H. Pottmann. Approximation of developable surfaces with cone spline surfaces. *Computer-Aided Design*, (30):571–582, 1998.

[19] Helmut Lütkepol. *Handbook of Matrices*. Jonh Wiley and Sons, 1996.

[20] Jean-Marie Morvan and Boris Thibert. Unfolding of surfaces. *Discrete and Computational Geometry*, 36(3):393–418, Oct 2006.

[21] James Munkres. *Topology (2nd Edition)*. Prentice Hall, 1999.

[22] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, August 2000.

[23] Boris Odehnal. Subdivision schemes for ruled surfaces. *Journal for Geometry and Graphics*, 12(1):35–52, 2008.

[24] Michael L. Overton. On minimizing the maximum eigenvalue of a symmetric matrix. *SIAM J. Matrix Anal. Appl.*, 9(2):256–268, 1988.

[25] Mathieu Perriollat and Adrien Bartoli. A single directrix quasi-minimal model for paper-like surfaces. *DEFORM'06 - Proceedings of the Workshop on Image Registration in Deformable Environments at BMVC'06*, pages 11–20, Sep 2006.

[26] Mathieu Perriollat and Adrien Bartoli. A quasi-minimal model for paper-like surfaces. *BenCOS07*, Jun 2007.

[27] Mathieu Perriollat, Richard Hartley, and Adrien Bartoli. Monocular template-based reconstruction of inextensible surfaces. *BMVC08*, 2008.

[28] Martin Peternell. Developable surface fitting to point clouds. *Comput. Aided Geom. Des.*, 21(8):785–803, 2004.

[29] J. Pilet, V. Lepetit, and P. Fua. Real-time non-rigid surface detection. In *Conference on Computer Vision and Pattern Recognition, San Diego, CA*, June 2005.

[30] J. Pilet, V. Lepetit, and P. Fua. Fast non-rigid surface detection, registration and realistic augmentation. *International Journal of Computer Vision*, 76(2), February 2008.

[31] Helmut Pottmann and Johannes Wallner. Approximation algorithms for developable surfaces. *Computer Aided Geometric Design*, 16:539–556, 1998.

[32] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[33] M. Salzmann and P. Fua. Reconstructing sharply folding surfaces: A convex formulation. In *Conference on Computer Vision and Pattern Recognition*, 2009.

[34] M. Salzmann, R. Hartley, and P. Fua. Convex optimization for deformable surface 3-d tracking. In *IEEE International Conference on Computer Vision*, Rio de Janeiro, Brazil, October 2007.

[35] M. Salzmann, J.Pilet, S.Ilic, and P.Fua. Surface deformation models for non-rigid 3–d shape recovery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(8):1481–1487, August 2007.

[36] Mathieu Salzmann, Francesc Moreno-Noguer, Vincent Lepetit, and Pascal Fua. Closed-form solution to non-rigid 3d surface registration. *ECCV*, 2008.

[37] Mathieu Salzmann, Raquel Urtasun, and Pascal Fua. Local deformation models for monocular 3d shape recovery. *CVPR*, 2008.

[38] Michael Spivak. *A Comprehensive Introduction to Differential Geometry*, volume 1–5. Publish or Perish, Inc., Berkeley, California, USA, 1975.

[39] Michael Spivak. *A Comprehensive introduction to Differential Geometry*. Publish or Perish, Inc, Berkeley, CA, USA, 1979.

[40] Meng Sun and Eugene Fiume. A technique for constructing developable surfaces. pages 176–185, 1996.

[41] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.