Optimization Methods for 3D Reconstruction

Depth Sensors, Distance Functions and Low-Rank Models

Bylow, Erik

2018

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Bylow, E. (2018). *Optimization Methods for 3D Reconstruction: Depth Sensors, Distance Functions and Low-Rank Models*. Department of Mathematics, Lund University.

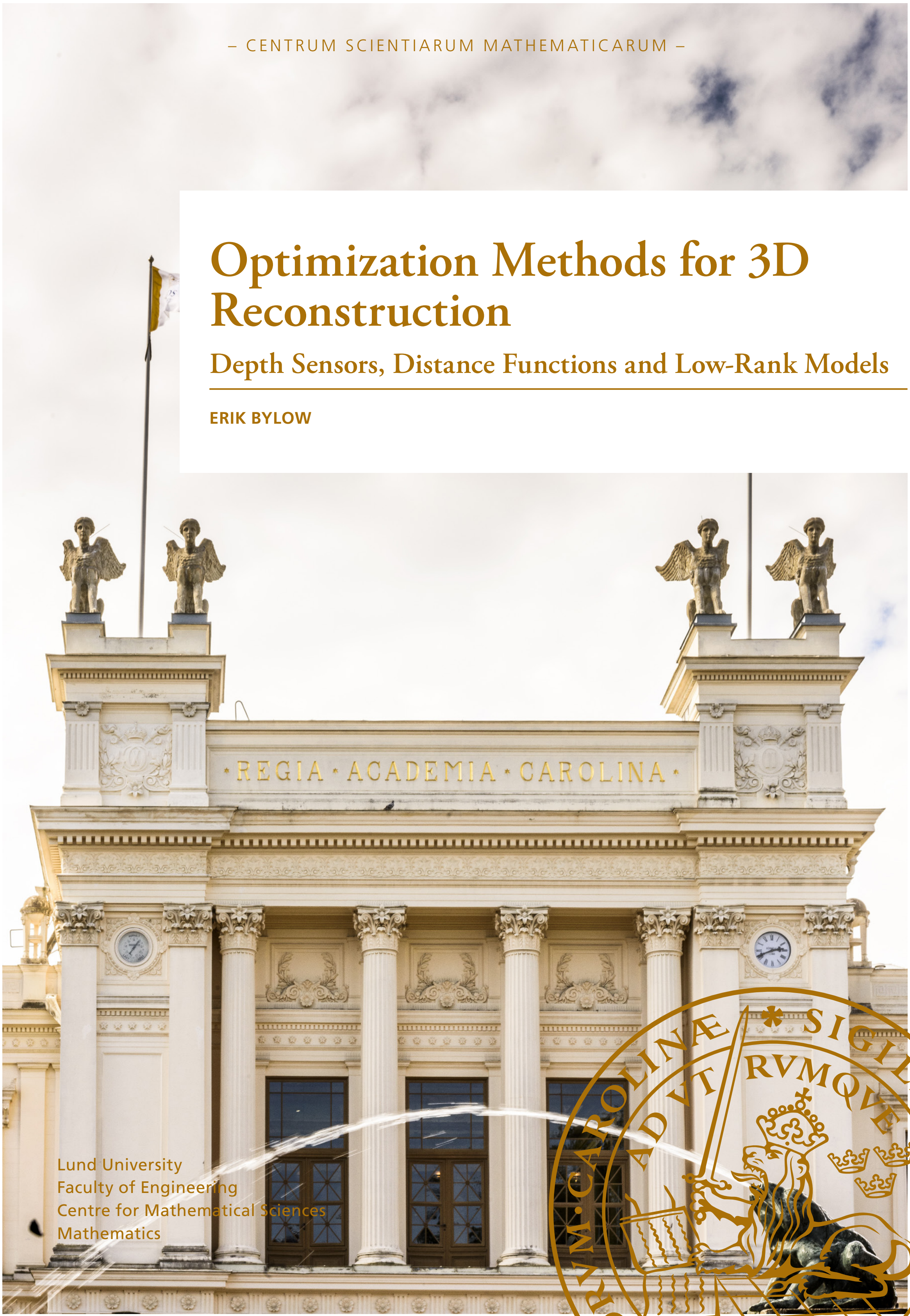
Total number of authors:
1

– CENTRUM SCIENTIARUM MATHEMATICARUM –

Optimization Methods for 3D Reconstruction

Depth Sensors, Distance Functions and Low-Rank Models

ERIK BYLOW



Lund University
Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Optimization Methods for 3D Reconstruction

Depth Sensors, Distance Functions and Low-Rank Models

Erik Bylow

ACADEMIC THESIS

which, by due permission of the Faculty of Engineering at Lund University, will be publicly defended on Friday 20th of April, 2018, at 13:15 in lecture hall MH:Hörmander, Centre for Mathematical Sciences, Sölvegatan 18, Lund, for the degree of Doctor of Philosophy in Engineering.

Faculty opponent

Dr. Christopher Zach, Toshiba Research Cambridge, United Kingdom.

Organization Centre for Mathematical Sciences Faculty of Engineering Lund Unversity Box 118 SE-221 00 Lund Author(s) Erik Bylow	Document name DOCTORATE THESIS IN MATHEMATICAL SCIENCES Date of issue March 2018 Sponsoring organization Supervisors Carl Olsson, Fredrik Kahl, Fredrik Andersson

Title and subtitle

Optimization Methods for 3D Reconstruction: Depth Sensors, Distance Functions and Low-Rank Models

Abstract

This thesis explores methods for estimating 3D models using depth sensors and finding low-rank approximations of matrices. In the first part we focus on how to estimate the movement of a depth camera and creating a 3D model of the scene. Given an accurate estimation of the camera position, we can produce dense 3D models using the images obtained from the camera. We present algorithms that are both accurate, robust and in addition, fast enough for online 3D reconstruction in real-time. The frame rate varies between about 5-20 Hz. It is shown in experiments that these algorithms are viable for several different applications such as autonomous quadrocopter navigation and object reconstruction.

In the second part we study the problem of finding a low-rank approximation of a given matrix. This has several applications in computer vision such as rigid and non-rigid Structure from Motion, denoising, photometric stereo and so on. Two convex relaxations which take both the rank function and a data term into account are introduced and analyzed together with a non-convex relaxation. It is shown that these methods often avoid shrinkage bias and give better results than the common heuristic of replacing the rank function with the nuclear norm.

Key words
Computer Vision, 3D Reconstruction, Low-Rank Models

Classification system and/or index terms (if any)

Supplementary bibliographical information		Language English
ISSN and key title 1404-0034		ISBN 978-91-7753-622-2
Recipient's notes	Number of pages xviii+122 Security classification	Price

Signature [illegible] Date 2018-03-06

Optimization Methods for 3D Reconstruction

Depth Sensors, Distance Functions and Low-Rank Models

Erik Bylow

Lund University

Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mathematics
Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden

http://www.maths.lth.se/

Doctoral Theses in Mathematical Sciences 2018:2
ISSN 1404-0034

ISBN 978-91-7753-622-2 (print)
ISBN 978-91-7753-623-9 (electronic)
LUTFMA-1066-2018

Printed in Sweden by MediaTryck, Lund 2018

Abstract

This thesis explores methods for estimating 3D models using depth sensors and finding low-rank approximations of matrices. In the first part we focus on how to estimate the movement of a depth camera and creating a 3D model of the scene. Given an accurate estimation of the camera position, we can produce dense 3D models using the images obtained from the camera. We present algorithms that are both accurate, robust and in addition, fast enough for online 3D reconstruction in real-time. The frame rate varies between about 5-20 Hz. It is shown in experiments that these algorithms are viable for several different applications such as autonomous quadrocopter navigation and object reconstruction.

In the second part we study the problem of finding a low-rank approximation of a given matrix. This has several applications in computer vision such as rigid and non-rigid Structure from Motion, denoising, photometric stereo and so on. Two convex relaxations which take both the rank function and a data term into account are introduced and analyzed together with a non-convex relaxation. It is shown that these methods often avoid shrinkage bias and give better results than the common heuristic of replacing the rank function with the nuclear norm.

Populärvetenskaplig sammanfattning

Datorseende och bildanalys är forskningsområden inom den tillämpade matematiken som de senaste åren blivit mer och mer vanligt förekommande i praktiska tillämpningar. Forskning inom datorseende går ut på att utnyttja information i bilder för olika tillämpningar. Detta gör man genom att utveckla för ändamålet lämpliga matematiska modeller.

Det finns idag många praktiska exempel där datorseende används. Exempelvis finns i varje smartphone en kamera med ansiktsigenkänning. En del telefoner klarar till och med av att göra 3D-modeller av människor. Ett område som är väldigt uppmärksammat just nu är självkörande bilar. Här finns flera olika metoder från datorseende såsom identifikation av människor och skyltar, beräkning av var bilen befinner sig och 3D-rekonstruktion av omgivningen.

Denna avhandling handlar om hur man utifrån bilder kan rekonstruera scener i 3D. I synnerhet studeras hur man med hjälp av djupsensorer kan skatta både kamerans rörelse och samtidigt skapa en rekonstruktion av vad kameran ser. En djupsensor är en typ av sensor som inte bara tar vanliga bilder, utan den mäter också avståndet mellan den observerade ytan och kameran, vilket kallas djupet. Den mest kända djupsensorn är förmodligen Kinecten som finns till Microsofts spelkonsol Xbox, men även Apple har precis släppt en djupsensor till sin nya iPhone. Tillgången till djupdata förenklar 3D-rekonstruktion jämfört med när man bara har tillgång till vanliga bilder.

Vill man göra en 3D-modell av ett objekt ska man ta bilder från olika vyer. 3D-rekonstruktion är ett typiskt "hönan eller ägget"-problem. Vet man kamerornas positioner kan man enkelt få fram en 3D-modell. Omvänt, om man vet 3D-modellen kan man enkelt bestämma kamerornas positioner.

Figur 1. En 3D-modell av författaren.

Här fokuserar vi på det fullständiga problemet att samtidigt bestämma både kamerornas positioner och 3D-modellen. Inom exempelvis robotik kan detta användas för självständig navigering och kartläggning av robotens arbetsmiljö. Man kan även generera 3D-modeller av till exempel människor, i Figur 1 syns en 3D-modell av författaren.

Rekonstruktion av objekt som är dynamiska och deformerbara är ett svårt problem som också studeras. Om man observerar ett objekt som rör på sig kommer punkterna på objektet röra på sig på olika sätt mellan varje bild. Tillåter man godtyckliga punktrörelser mellan varje bild är problemet illa ställt. Därför begränsar vi oss till rimliga/enkla rörelser. För sådana finns ofta mycket beroende mellan de observerade punktrörelserna. Ett exempel är en människas arm. Om man följer punkter på både överarmen och underarmen så kan dessa röra sig på olika sätt. Däremot kan de inte röra på sig hur som helst i förhållande till varandra, punkter som tillhör underarmen kommer röra sig väldigt likartat och likaså punkter på överarmen. Det finns även en koppling mellan hur punkterna på underarmen rör sig i förhållande till överarmen. Vi letar alltså efter lösningar med starka beroenden mellan punktrörelser.

De observerade punkterna sparar vi i kolonner i en matris, där varje kolonn beskriver en specifik punkts rörelse i alla bilder. Med matematiska termer beskrivs styrkan i beroendet av den så kallade rang-funktionen. Ju högre rang, desto mer komplex rörelse tillåts och beroendet mellan punktrörelserna blir svagare. För att få en realistisk rekonstruktion söker vi därför hitta lösningar med låg rang som stämmer väl med observationerna. På grund av att rang-funktionen inte är kontinuerlig är de optimeringsproblem som uppkommer svåra att lösa. I den här avhandlingen presenterar vi ett antal approximationer och förenklingar som ger bra resultat och samtidigt är lätta att hantera.

Preface

"I think you will end up with a doctor's hat on your head". That was one of the last things my grandfather told me the last time I met him in September 2010. Now I can present my Doctoral Thesis. The contents of the thesis is based on the following papers:

Main papers

- Real-Time Camera Tracking and 3D Reconstructions Using Signed Distance Functions, E. Bylow, J. Sturm, C. Kerl, F. Kahl, D. Cremers, *Robotics: Science and Systems*, Berlin, 2013.

  Contributions: J.S. came up with the idea. I developed the theory and did all implementation and wrote most of the paper with input from the other authors. C.K. and J.S. did the quadrocopter experiments and I did the remaining experiments.

- Robust Camera Tracking by Combining Color and Depth Measurements, E. Bylow, C. Olsson, F. Kahl, *Inernational Conference on Pattern Recognition*, Stockholm, 2014.

  Contributions: I came up with the idea and developed the method together with input from CO and FK. The implementation and all experiments were done by me. The paper was written by me with input from CO and FK.

- Rank Minimization With Structured Data Patterns, V. Larsson, C. Olsson, E. Bylow, F. Kahl, *European Conference on Computer Vision*, Zürich, 2014.

  Contributions: VL and CO developed the main theory with input from the co-authors. The experiments were mainly made by CO, VL and me.

VL wrote the paper with input from the co-authors.

- Robust online 3D reconstruction combining a depth sensor and sparse feature points, E. Bylow, C. Olsson, F. Kahl, *International Conference on Pattern Recognition*, Cancún, 2016

  Contributions: I came up with the idea, developed the theory with input from CO and FK. The implementation of the algorithm was done by me. The paper was written by me with input from CO and FK.

- Minimizing the Maximal Rank, E. Bylow, C. Olsson F. Kahl, M. Nilsson, *Conference on Computer Vision and Pattern Recognition*, Las Vegas 2016

  Contributions: CO and I developed the theory. I did the experiments and wrote the paper with input from the co-authors.

- Dense tracking and mapping with a quadrocopter, J. Sturm, E. Bylow, C. Kerl, F. Kahl, D. Cremers, *International Conference on Unmanned Aerial Vehicles in Geomatics*, Rostock, 2013.

  Contribution: This is an extension of the *RSS* paper. My contribution was to develop the method and writing most of the code. The experiments were performed by JS and CK. The paper was mostly written by JS with input from the co-authors.

- CopyMe3D: Scanning and printing persons in 3D, J. Sturm, E. Bylow, F. Kahl, D. Cremers, in Proc. *German Conference on Pattern Recognition*, Saarbrücken, 2013.

  Contribution: This is another extension of the *RSS* paper. My contribution was developing the basic code. JS adjusted it for the particular application and performed the experiments. The paper was written by JS with input from the co-authors.

- A Non-Convex Relaxation for Fixed-Rank Approximation, C. Olsson, M. Carlsson, E. Bylow, *RSL-CV (International Conference on Computer Vision - workshop)*, Venice 2017

  Contribution: CO and MC developed most of the theory. I helped CO with the main result and contributed to the writing.

Subsidiary papers

- Direct camera pose tracking and mapping with signed distance functions, E Bylow, J Sturm, C Kerl, F Kahl, D Cremers, *4th Workshop on RGB-D: Advanced Reasoning with Depth Cameras (Robotics: Science and Systems-workshop), Berlin 2013*

Acknowledgements

I would like to thank my colleagues at the Centre for Mathematical Sciences and in particular the Computer Vision Group for all their support and help. I want to give a special thank to my supervisors Carl Olsson and Fredrik Kahl for fruitful discussions and help in general with all types of problems. Also I would like to thank Daniel Cremers at the Technichal University of Munich whom I visited both as a Master Student and also during my time as a PhD. I would also like to thank Jürgen Sturm for his support in the beginning of my research career.

Last but not least I want to thank my family for their support during all times. I want to give a special thought to my late mother and grandfather, who both always supported me and my sisters' studies.

This work has been funded by the Swedish Research Council (grants no. 2012-4213, 2012-4215) and the Crafoord Foundation.

Contents

Chapter 1

Introduction

Computer vision is a broad research area with several different subfields. Each field has its own focus and applications, but all have in common that they are working with images. This thesis studies mainly how 3D models can be created using special cameras known as depth sensors and how a matrix with data can be approximated by a matrix of low rank. The latter problem occurs in several applications of computer vision, for example, in Structure from Motion and image denoising. The former topic is less abstract and there are several applications for 3D models. The models themselves have a value and can for example be used in refurbishment. Also in for example robotics one can have the application that the robot shall navigate on its own in a room. With an estimation of the scene and the current position of the camera, this information can help the robot to perform its task.

The problem of approximating an observed matrix with a matrix of low rank has applications in computer vision as well. For example, image based 3D reconstruction can be done using this technique. Also photometric stereo and denoising are problems that can be solved by approximating a given measurement matrix with one of low rank.

1.1 Organization of the Thesis

Chapter 2 In this chapter we give a brief introduction to rigid 3D reconstruction. This chapter contains basic information about the pinhole camera model, depth images, surface representations and other basic concepts intended for the reader who is not familiar with the topic.

Chapter 3 In this chapter the focus is on online camera pose estimation, which means that the 3D model and the camera pose are estimated as images are captured. We study how one can use the 3D model itself to estimate the pose of the camera by using a signed distance function. Results suggest that using the model directly to estimate the pose works better than Iterated Closest Points, ICP. This chapter is based on the papers [13, 68, 67].

Chapter 4 In this chapter we improve the camera pose estimation to handle more general environments. This is done by studying how color information from the model can be invoked in the tracking as well as how feature points from the images can be used. This chapter is based on the papers [14, 15].

Chapter 5 In this chapter we tackle the problem of approximating a measurement matrix with one of low rank. We show how to derive a convex envelope of our objective that consists of a penalizing rank term and a data term. In particular we compute the bi-conjugate of our objective which can then be combined with other convex constraints. This chapter is based on [45].

Chapter 6 Here we derive another method for low-rank approximation. We focus on estimating several matrices at the same time and favor solutions where all matrices have the same rank. This is a natural formulation in applications like image denoising. This chapter is based on [16].

Chapter 7 In this chapter we are looking for a solution to a linear system with the rank-function as regularization. We show that if the linear operator $\mathcal{A}$ fulfills the Restricted Isometry Property, RIP, we can give optimality guarantees. This chapter is based on [53].

Chapter 2

Rigid 3D Reconstruction

2.1 Introduction

The capability to reconstruct a scene from a set of images has been one of the major research topics in computer vision. It is still an active research area and many challenges remain. This research field is known as Structure from Motion (SfM) and in robotics as Simultaneous Localization and Mapping (SLAM). Sometimes a distinction is made between SLAM and SfM. Then SfM typically refers to being offline in the sense that all images are captured before the reconstruction and pose estimation are taking place. SLAM would then correspond to estimating the camera pose and 3D scene as the images are captured.

In SfM, the pipeline is typically to first find a set of key points in different images and to match these. Then one computes the camera poses and 3D scene geometry using bundle-adjustment [32] to get global consistency, meaning that the 3D points computed from different views align in the resulting 3D model. This results in a sparse 3D model, where only the detected key points can be reconstructed.

It is also possible to create dense 3D models using monocular cameras. One approach is to use stereo and try to estimate depth maps from image pairs. Each depth image is created by optimizing an energy function for each image pair where the goal is to find the disparity for each pixel, which is inversely related to the depth. Often these energy functions require some form of regularization to decrease noise and to get smooth surfaces.

Today there is a new type of cheap sensors available on the market which can also measure distances. These are known as depth sensors and the first commercial one was probably the Kinect camera for Xbox 360. The key feature with these sensors is the capability to generate depth images which measures the metric distance between the sensor and the object. Therefore, one gets information about both scale and the scene in 3D without any explicit computations. The range for these sensors typically lies between 0.8 - 4.0 m. Most depth sensors rely on structured light which might not work outdoor. However, for smaller indoor scenes these sensors generate dense depth images which can be used to create 3D models. The focus of the first part of the thesis is how to estimate the scene and camera pose in a robust and accurate way.

The motivation behind this is that an accurate and simultaneous estimation of the 3D model as well as the camera pose is needed in several applications. For instance in robotics, the robot can use the current position and 3D model to localize itself in the room. If the camera position is poorly estimated, then the robot will make an incorrect estimation of where it is and might not be able to perform its task. In other applications such as refurbishment, one might want to measure the dimensions of a room. This could then for example be used to see how a new sofa would fit into a living room. To get good measurements of the dimensions of the sofa an accurate model is needed. Also the gaming industry can make use of acquired 3D models and it can potentially also be used in augmented- and virtual-reality.

The main contribution of the first part of the thesis is that we develop robust and accurate methods for tracking the pose of the depth sensor in real-time. By real-time we mean that the pose is estimated and the model is created as we acquire new depth- and color-images, in contrast to offline methods which first capture all frames before performing pose estimation and 3D reconstruction.

2.2 Basics

In this section concepts from computer vision are presented. We will describe the pinhole camera model and illustrate the idea behind some different well-known methods for computing the camera pose and representing the 3D model.

2.2.1 Pinhole Camera Model

The pinhole camera model is probably the most common model of a camera. Let $\mathbf{x}$ be a point in the world, seen by the camera. Then under the pinhole camera model the point is projected onto the image plane by following the ray between the point $\mathbf{x}$ and the camera center $\mathbf{c}$. The projected point is the intersection between the image plane and the ray, as illustrated in Figure 2.1. The projection on the image plane for a 3D point $\mathbf{x}$, represented by coordinates (x, y, z) is

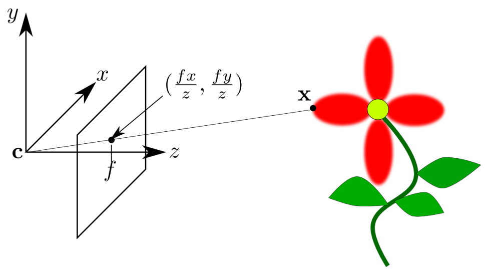


Figure 2.1: A 3D point $\mathbf{x}$ is projected onto the image plane by $(\frac{fx}{z}, \frac{fy}{z})$, where f is the distance between the image plane and the camera center $\mathbf{c}$.

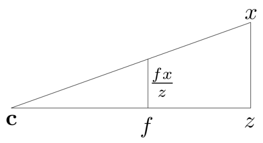


Figure 2.2: If a 3D point has the x-coordinate x and z-coordinate z, then the projection on the image plane is given by $\frac{fx}{z}$.

$(\frac{fx}{z}, \frac{fy}{z}, f)$ where the image plane is located in front of the camera at distance f, which is the focal length. This is visualized by similar triangles in Figure 2.2. We assume that the image plane is parallel with the xy-plane and we are only interested in the coordinates $(\frac{fx}{z}, \frac{fy}{z})$ on the image plane, as illustrated in Figure 2.1. To change origin in the image coordinates we simply translate the projected point $(\frac{fx}{z}, \frac{fy}{z})$ by (c_x, c_y), where c_x is half the width of the image plane in pixels and c_y half the height in pixels. To get the pixel coordinates of the 3D point $\mathbf{x}$ we compute

$$(p_x, p_y) = (\frac{fx}{z} + c_x, \frac{fy}{z} + c_y). \tag{2.1}$$

We now define a function that takes a 3D coordinate to pixel coordinates:

Definition 2.2.1. Let $\pi : \mathbb{R}^3 \to \mathbb{R}^2$ be the function that takes a 3D point to pixel coordinates:

$$\pi(\mathbf{x}) = (\frac{fx}{z} + c_x, \frac{fy}{z} + c_y). \tag{2.2}$$

The parameters f, c_x and c_y are known as the intrinsic camera parameters and if we know them we say that the camera is calibrated.

When we have the image projections of a 3D point, but not the 3D point itself, we may want to do the reverse. 3D reconstruction can be thought of as inverting the projection. Assume that we know the projections and the calibration of the camera. How do we get back the original 3D point? We know that the 3D point must lie on the ray between the pixel and the camera center so we can simply do the reverse calculations and get

$$\begin{aligned} x &= \frac{p_x - c_x}{f} t \\ y &= \frac{p_y - c_y}{f} t \\ z &= t, \end{aligned} \tag{2.3}$$

where $t \in \mathbb{R}^+$ is an arbitrary scalar corresponding to the depth. However, we have an ambiguity here because we can take any depth t and we will get a 3D point that projects onto the same pixel coordinates (p_x, p_y).

This ambiguity is resolved if we are using depth sensors. That is because for each pixel the sensor estimates the distance to the object, consequently we get a measurement z at pixels (p_x, p_y). Therefore, we can replace the unknown t with the known z in (2.3).

For each camera, a point cloud can be created by performing the calculations in (2.3) for each pixel. However, point clouds observed from different views will not align well unless we know where the camera was when the different images were captured. What we need to know is how the camera was rotated and translated with respect some global frame of reference. The information we need is thus the rotation R and translation $\mathbf{t}$. By doing the computations in (2.3) we get a local 3D point $\mathbf{x}_L$. To transform this to the global frame of reference we compute

$$\mathbf{x}_G = R\mathbf{x}_L + \mathbf{t}. \tag{2.4}$$

2.2.2 Affine Camera Model

An affine camera model is an approximation of the projective camera. In Sectione 2.2.1 we saw that the projection of a point $\mathbf{x}$ was the intersection of the

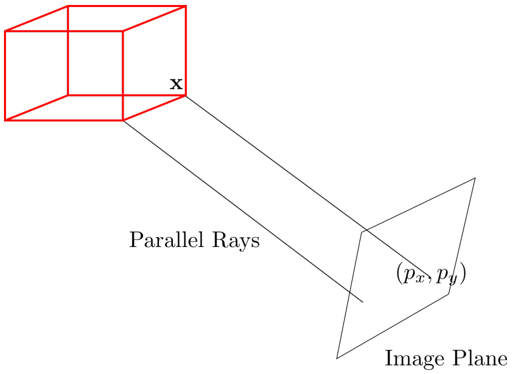


Figure 2.3: An affine camera model, all 3D points are projected parallel to the image plane. In other words, the camera center is at infinity where the parallel rays meet.

ray between $\mathbf{x}$ and the camera center and the image plane. In the affine camera model, the rays intersecting the image plane are assumed to be parallel. The camera model looks like

$$\begin{pmatrix} A & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \tag{2.5}$$

where $A \in \mathbb{R}^{2\times 3}$, $\mathbf{t} \in \mathbb{R}^2$ and A has rank 2. If we have a regular 3D point $\mathbf{x}$ and project it onto the image plane using homogeneous coordinates we get

$$\begin{pmatrix} A & \mathbf{t} \\ \mathbf{0} & 1 \end{pmatrix} \begin{pmatrix} \mathbf{x} \\ 1 \end{pmatrix} = \begin{pmatrix} A\mathbf{x} + \mathbf{t} \\ 1 \end{pmatrix} . \tag{2.6}$$

Since the last row will always be one when we use regular 3D points the projection simplifies to

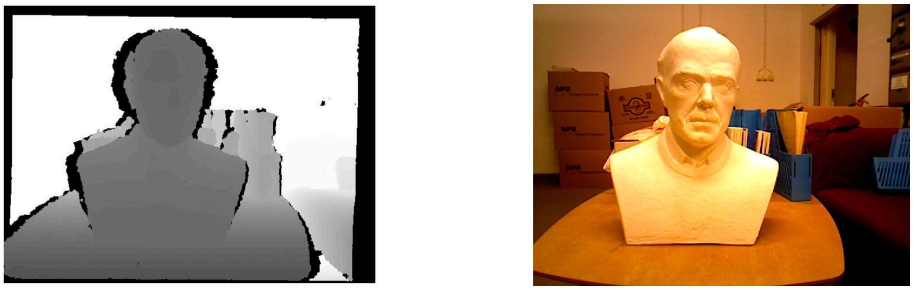
$$A\mathbf{x} + \mathbf{t} = \begin{pmatrix} p_x \\ p_y \end{pmatrix} . \tag{2.7}$$

The consequence of this is that in the cameras local frame of reference the 3D point $\mathbf{x}$ has the same x- and y-coordinates as the projection $(p_x, p_y)^T$. An illustration of the projection is shown in Figure 2.3. In scenes where the 3D points lie at different depths this model is a poor estimation of the image formation.

However, if the 3D points lie far from the camera and all 3D points have roughly the same depth the approximation can be good. More information about affine cameras and other camera models can be found in [32, 29]

2.2.3 Depth Cameras

Depth cameras differ from regular cameras in the sense that they generate two different images, one color image and one depth image. Each pixel in the depth image contains distance information between the object and the camera. In Figures 2.4a and 2.4b a depth image with its corresponding color image are shown.



(a) An example of a depth image

(b) Corresponding color image.

Figure 2.4: An example of an image pair that the depth sensor provides, a depth image (*left*) and a color image (*right*).

Let us denote the depth image by I_d, then we can for each pixel (p_x, p_y) read the depth value

$$z = I_d(p_x, p_y). \tag{2.8}$$

Using this we resolve the ambiguity of the depth in equation (2.3). We define the following function that maps a pixel (p_x, p_y) to its 3D coordinates:

Definition 2.2.2. Let $\rho : \mathbb{R}^2 \times \mathbb{R} \to \mathbb{R}^3$ be the function that transforms a pixel (p_x, p_y) to its 3D coordinates by

$$\rho(p_x, p_y, z) = (\frac{p_x - c_x}{f} z, \frac{p_y - c_y}{f} z, z), \tag{2.9}$$

where $z = I_d(p_x, p_y)$ and f, c_x and c_y are the intrinsic camera parameters.

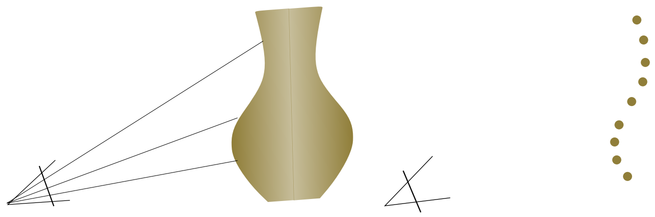


Figure 2.5: In the image to the left the depth sensor captures a depth image of the vase. To the right we can see how the point cloud could look like by reconstructing the 3D points.

An illustration of how this works is given in Figure 2.5, where the points to the right are reconstructed 3D points of the vase.

2.2.4 Representation of 3D Models

Using the depth image alone, we get a point cloud. However, we would like to estimate a surface, not just a set of points. There are different ways of representing 3D models. What one typically wants is a method to represent the 3D model that it is memory efficient, fast and flexible. By flexibility we mean that there are no constraints on what the model should look like, instead we want to be able to reconstruct any object. Here we present two popular methods when working with depth sensors. Both have advantages and disadvantages but they can represent any topology of the surface which is a major advantage.

Octrees

One way of representing the surface is to use a probabilistic occupancy grid as in [73], where the space is represented via an octree. An octree is a memory efficient tree data structure where each parent has exactly eight children. In 3D one parent would correspond to a cube in space, which can be partitioned into eight new cubes, where each new cube corresponds to a child of the parent, as depicted in Figure 2.6. This allows for having a high resolution close to the surface and lower resolution where there is no surface.

The key idea is to divide the space into cells and label a cell occupied if it contains a surface point. If there is no surface point the cell is labeled free. Since this approach is very memory efficient, it is possible to do large scale representations.

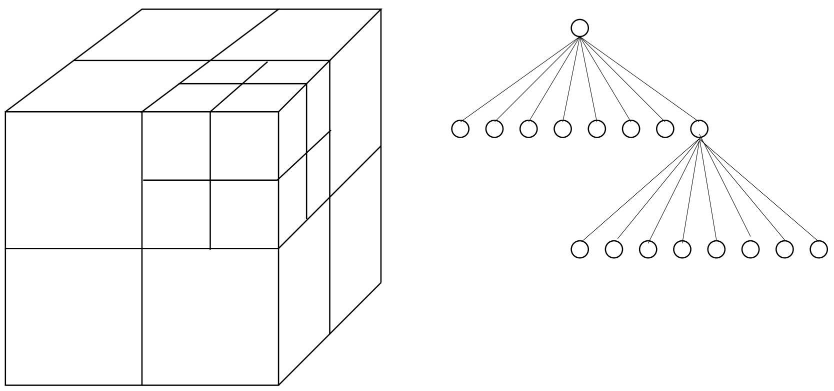


Figure 2.6: In an octree, each parent has eight children, which can be used to subdivide space with higher resolution in certain parts.

The fact that both free- and occupied-space is represented can be used to avoid obstacles in for example robotics.

Signed Distance Functions

An alternative representation, which is commonly used in conjunction with RGB-D cameras, [71], [37], [50], [64], is to use a so called Truncated Signed Distance Function, TSDF.

Let us start with a basic example of how signed distance functions (SDF) works. Assume that we have the following function:

$$f(x, y, z) = 1 - \sqrt{x^2 + y^2 + z^2}. \tag{2.10}$$

If we choose (x, y, z) such that $f(x, y, z) > 0$, the point lies inside the sphere, or if we take (x, y, z) such that $f(x, y, z) < 0$, then we are outside the sphere. Obviously, the surface of the sphere lies at the zero level of the function f. This is thus an implicit representation of the surface. This is also illustrated in 2D in Figure 2.7. The figure contains a red area, a blue area and a white circle between the two areas. This is an implicit representation of a circle where points outside the circle have a negative function value which is the distance and are colored blue. The points inside the circle have a positive distance to the surface and are colored red. The theory for level set methods and surface representations through signed distance functions is thoroughly treated in [54].

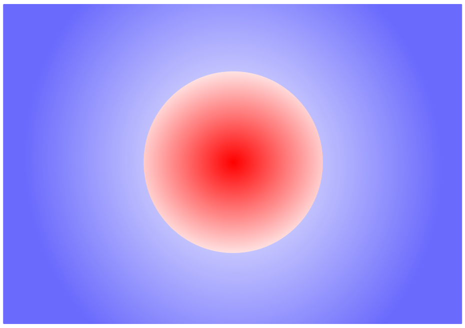


Figure 2.7: A circle (white) is represented implicitly through a signed distance function. The points with red color are inside the circle, having a positive distance and the blue points are points outside the circle and they have a negative distance

The obvious drawback of this representation is that one needs to estimate the distance to the surface for each point in space. For a general 3D model, no known closed form solution exists.

To achieve an approximation of the signed distance function, we use a uniform voxel grid. A voxel is like a pixel but in 3D which contains data and has a fixed position in space. In this case, we want to estimate the distance between the surface and the voxel and store it.

To estimate the distance to the surface for each voxel one often computes a weighted average of the measurements from different views. This has the effect that noise can be averaged out and one can often obtain smooth surfaces. Another advantage is that there is no restriction on what the 3D model might look like and it is easy to parallelize computations since many operations on the voxel nodes are independent of its neighbors. A drawback with the uniform representation is that it requires a lot of memory.

The challenge to estimate the distance between the voxels and the surface remains. In Chapter 3 we present a method for how to estimate the signed distance function.

In this work we will not be using signed distance functions directly, but instead we will use a TSDF instead. This is an approximation of the SDF with the restriction that the distance is only estimated in the vicinity of the surface up to a threshold δ. Voxels that are far from the surface get the distance estimation truncated to δ. In principle the idea is the same, the surface is represented implicitly

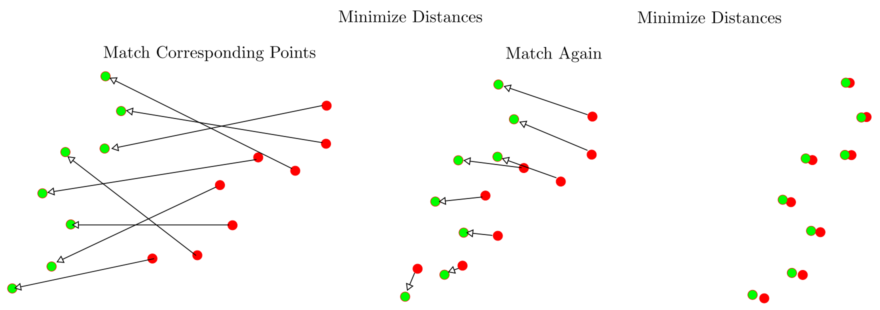


Figure 2.8: In ICP we have two point clouds and the goal is to find a transformation that takes the red point cloud to the green point cloud in an optimal way. One alternates between finding correspondences and minimizing the distance between these correspondences.

through the zero-level. More information about TSDF:s can be found in [22].

2.2.5 Camera Tracking

To align point clouds from different views, the global rotation R and translation $\mathbf{t}$ have to be known. This is also known as the pose of the camera. The problem of estimating the pose of the camera has been studied for a long time, [9, 34, 10, 21]. Here two methods are described in order to give a basic understanding of how they work.

Iterated Closest Point

In the beginning of the 1990, [9] was published. The paper describes a method for how to register two point clouds. The technique is known as *Iterated Closest Point*, (ICP). The method has become a standard approach for 3D registration and there are almost infinitely many versions of it, a survey can be found in [61]. Given two point clouds, we want to align them and ICP aims to find a rotation and translation which transforms a given point cloud into the other by minimizing the sum of distances in some norm, often L_2, as illustrated in Figure 2.8.

From Figure 2.8 we can extract some key components of ICP. Given two point clouds with points $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ (green points in Figure 2.8) and corresponding points $(\mathbf{y}_1, \ldots, \mathbf{y}_n)$, (red), we aim to minimize

$$E(R, \mathbf{t}) = \sum_{i=1}^{n} \|R\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i\|^p \tag{2.11}$$

which means we are seeking the rotation and translation that minimizes the sum of the residuals, often with $p = 1$ or $p = 2$.

With no noise and perfect correspondences this is easily solved [34] if $p = 2$. However, given two point clouds, finding good correspondences is a problem in its own. Typically we cannot expect to find correspondences for all points, but just for a subset of points and not all of these might be correct. To handle this we start with a set of initial correspondences, then solve (2.11). Then we can apply the transformation on the point cloud and recompute the correspondences and solve (2.11) again, until convergence. This is local optimization with no guarantee of finding the global optimum. However, if the relative rotation and translation between the two cameras are small, the chance for finding the correct pose is higher.

There are several variants of this procedure, a common method is to estimate a normal to each point and minimize the projection on the normal instead of the difference between the point pairs, [21]. We would then minimize

$$\sum_{i=1}^{n} |(R\mathbf{x}_i + \mathbf{t} - \mathbf{y}_i)^T \mathbf{n}_i|^p. \tag{2.12}$$

This metric is known as the point-to-plane metric and corresponds to minimizing the distance between the point $\mathbf{x}_i$ and the tangent plane at $\mathbf{y}_i$.

A general drawback of ICP is that most methods rely on the presence of varied geometry. If all points lie on a plane, there is no unique minimizer of (2.11) or (2.12).

Intensity Based Methods

Some other approaches which work well and have some nice properties have also been developed. One approach is to instead of minimizing the sum of residuals between two point clouds, the projected photo-consistency between two color images should be maximized. This relies on having corresponding depth images. That is, if a 3D point for a pixel in one image is projected onto another image,

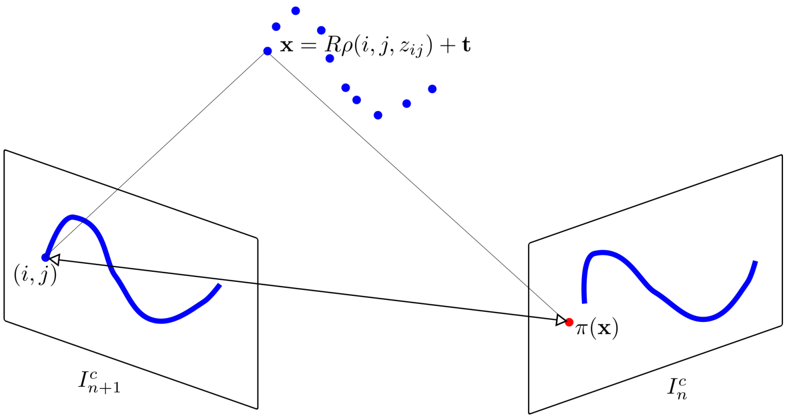


Figure 2.9: A pixel (i, j) with depth z_{ij} is reprojected to 3D as $\mathbf{x} = R\rho(i, j, z_{ij}) + \mathbf{t}$ and then projected on the image I^n_c and pixel coordinates $\pi(\mathbf{x})$. One then compares the intensity values at pixel (i, j) in I^{n+1}_c and $\pi(\mathbf{x})$ in I^n_c. In this case intensities will not match.

then the intensity in these two pixels should be similar. We call the difference in intensity between these pixels the intensity error. In particular [63] introduced a frame-to-frame tracking approach which uses both the depth image and the RGB image. This was later extended in [64] to handle more general situations.

Intensity based methods such as this are suitable for RGB-D cameras where you have a stream of images (I^n_d, I^n_c). The idea is to create a point cloud from one image. Then we project the points into the second image. The rotation R and translation $\mathbf{t}$ is assumed to be correct if the intensity difference is 0. In practice perfect color matching cannot be achieved. Therefore the intensity difference is typically minimized instead.

This differs from ICP in that we are not aligning two point clouds, but instead we are trying to maximize photo consistency. As illustrated in Figure 2.9, we have a 3D point $\mathbf{x}$ obtained from the left image. Then we use the estimated relative transformation between camera position C^n and camera position C^{n+1} and transform $\mathbf{x}$ to the second camera frame. The point is then projected onto pixel $\pi(\mathbf{x})$ in I^n_c and the difference in intensity between pixel (i, j) in I^{n+1}_c and $\pi(\mathbf{x})$ in I^n_c is evaluated. This process can be formulated as the following energy function:

$$E(R, \mathbf{t}) = \sum_{i=1}^{m} \sum_{j=1}^{n} \| I_c^{n+1}(i, j) - I_c^n(\pi(R\rho(i, j, z_{ij}) + \mathbf{t})) \|^2,$$

where z_{ij} is the depth at pixel (i, j) in the depth image $I_d^n(i, j)$ observed at time step k, m is the number of rows and n is the number of columns in the image. With this technique we find the relative transformation between two frames.

The advantage of this approach is that we can use all color information available and need not to find corresponding points as in ICP. The disadvantage is that errors are quickly accumulated since the tracking is frame-to-frame. This can lead to poor results if we do not find a way of reducing these errors.

Chapter 3

Estimating the Camera Pose and Creating a 3D Model

3.1 Introduction

The main contribution with this chapter is that we investigate how the 3D model represented as a Truncated Signed Distance Function, TSDF, can be used to estimate the camera pose. Evaluation on benchmarks demonstrates that when using the 3D model in a different way than KinectFusion [50, 37] does, the estimated pose is even more accurate. These conclusions are drawn by evaluating an open-source implementation of KinectFusion, known as KinFu [1], together with our method on publicly available datasets [66].

As described in Chapter 2.2, there are some different ways of creating 3D models and estimating the camera pose. For applications like robotics, refurbishment, virtual reality and so forth, it is important to have a method that can estimate both the pose and the 3D model accurately and robustly. Here we will see that by using information in the TSDF, the pose of the camera can be robustly estimated in an online manner, which means that the camera pose and the 3D model are estimated as images are captured. With an accurate estimation of the camera pose, the 3D model will also be of good quality, as will be seen in the experiments.

3.1.1 Related Work

As mentioned earlier, 3D reconstruction is not a new topic and there are many approaches to solve the problem of estimating the pose and the geometry. Both in SfM and in SLAM a lot of research have been done. Lately, since the advent of the Microsoft Kinect and the Asus Pro Live sensor, a very active research area has

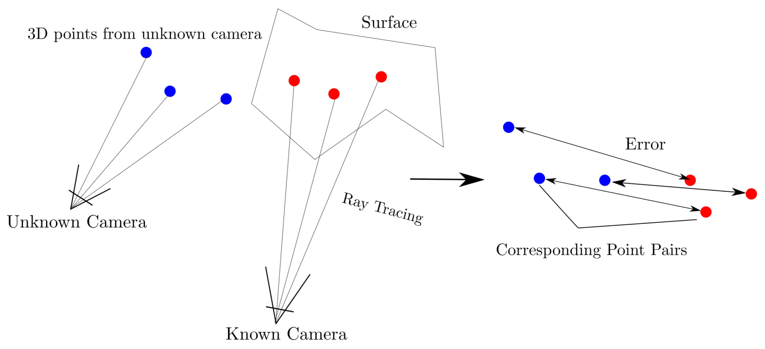


Figure 3.1: Given a new depth image, one takes the last known camera position and perform ray tracing to find points on the actual surface. With this global point cloud one performs ICP to find the correct transformation between the last known camera position and the new unknown camera position.

been how to create 3D models using depth cameras.

The most famous work is probably KinectFusion [50], which was perhaps the first system capable of creating 3D models in real-time using these sensors. Their main contribution was that it was demonstrated how a TSDF can be used to robustly track the camera movement for medium-sized reconstructions. To track the camera ICP is used and to create the 3D models, the method from [22] is used. What makes ICP more robust in KinectFusion is that the 3D model is rendered directly. This results in two point clouds, one from the model and one from the new depth image. These are then aligned to each other which gives an estimation of the camera pose. An illustration of this is shown in Figure 3.1.

In [19] an algorithm similar to our is presented. The focus in [19] is however more on object detection and recognition in a TSDF and no evaluation is provided. Also [5] develops a similar algorithm, but test it only on synthetic data and no evaluation is provided. In [59] the same energy function as our is presented, however, a pre-computed SDF is assumed and no real-time 3D reconstruction is performed. Instead it is shown that the camera can be calibrated with known SDF.

A different approach to estimate the camera pose compared to KinectFusion is taken in [63]. Instead of minimizing the geometric error between point clouds, one seeks to maximize photo-consistency between two consecutive images as described in Section 2.2.5. Here the pose estimation is independent of the model.

Actually no model is estimated in [63]. This idea was improved in [40] and [41] where also consistency between the depth images is used. A drawback is that these methods easily drift away. Drifting means that due to accumulated errors, the estimated camera pose deviates more and more from the true path. These intensity based methods do not require a 3D model to work, which makes it possible to use other methods to decrease drift such as loop-closure and bundle-adjustment.

Another way to estimate the camera pose is the one presented by [26], where corresponding key points are found between pairs of images. Then RANSAC is used to compute a relative transformation between the image pair. These transformations are then added to a graph which optimizes the camera pose globally to reduce drift. This method has the advantage compared to common ICP that it requires only key points to be found, so it can handle scenes with little texture as long as it can find key points. A drawback is that the 3D model cannot be created until the entire scene is recorded. To represent the 3D model, a probabilistic Octree, [73] is used, which is created after all images have been recorded and the graph is globally optimized using g2o [43]. The advantage with such a representation is that it is very memory efficient. This is useful in robotic applications since both occupied and free space is represented.

The approaches described above have been successfully used in other well-known methods such as Kintinuous [71] and lately [72]. A combination of the KinectFusion based ICP [50] and the intensity based methods by [63] is used to estimate the camera trajectory. The aim of these works is to create large-scale online reconstructions, that is, the model is created as the images are recorded. For large-scale methods one faces other challenges. Firstly, one must reduce drift and secondly the memory consumption must be limited so that the entire model can be represented on a computer. To handle this [71] uses a rolling volume. This means that one has a uniform grid as in KinectFusion and represents the surface with a TSDF. However, when reaching the border of this grid the other half is saved to the hard drive and a new empty grid is appended to the volume where the camera is, as depicted in Figure 3.2.

In [72] a new interesting approach is taken. Instead of using a TSDF, the surface is represented by using surfels [57]. Surfels are basically small surface elements that contain information about position, size, orientation and possible texture. This makes it easier to recompute the 3D model online if drift is detected and adjusted for which is done in [72] with impressive results. To track the camera the KinectFusion based ICP is used together with an error term that takes photo

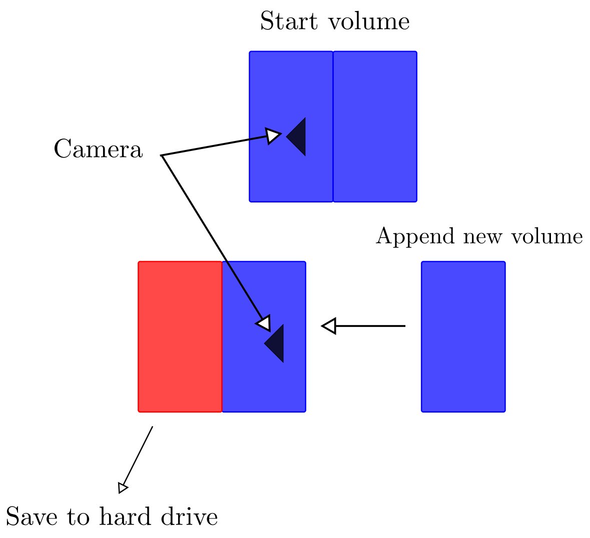


Figure 3.2: When the camera approaches the border of the volume, a new volume is created and appended to the active part. The other half is saved to the hard drive.

consistency into account. This makes it robust to scenes where there is either only texture or structure.

Another work that has shown impressive results and manages to correct the surface in an online manner is [24]. A TSDF is used to represent the model, but a smart and efficient implementation with the use of key frames allows for recomputing the surface in real-time. Bundle adjustment is continuously used to reduce drift.

3.2 Updating a TSDF for a New Depth Image

As described in the Chapter 2.2, we can use a TSDF to represent the 3D model. Here we go into detail of how the grid is updated as we get new measurements for each new frame.

In a true signed distance function ψ, for each point $\mathbf{x}$ we shall get the closest distance to the surface with sign if we evaluate $\psi(\mathbf{x})$. Also from the definition, see [54], we have the constraint $\|\nabla\psi\| = 1$ shall be fulfilled. Here we aim to do real-time 3D reconstruction similar to KinectFusion, so both constraints above will be relaxed. Since the Kinect delivers images at a rate of 30 frames per second, one has approximately 33 ms to find the position of the camera and update the grid with new information. To our knowledge, there is no easy and efficient method of estimating the SDF exactly. Instead we follow the heuristic by [22], which is trivial to parallelize, allowing for a considerable speed-up using a modern GPU.

To start with we have a voxel grid, which is a 3 dimensional discretization of a volume in space. Each voxel has a unique index $(i, j, k) \in \mathbb{N}^3$ and we refer to one voxel at index (i, j, k) as V_{ijk}. Each voxel stores data used to represent the distance function. The data in this work is:

- D - estimated signed distance to surface
- W - estimated weight of the measured distance
- R - estimated intensity in the red channel
- G - estimated intensity in the green channel
- B - estimated intensity in the blue channel.
- W_c - estimated weight for the color.

A data value for a voxel at (i, j, k) will be referred to with subscript ijk, for example the distance at voxel V_{ijk} will be denoted D_{ijk}. We set the origin of the global coordinate system to be in the center of the voxel grid. Since the distance between the voxels is known and the voxels are fixed in space and the origin of the global coordinate system is known, the coordinates for a voxel V_{ijk} can easily be computed.

With the above definitions we can now describe how we can estimate the signed distance to the surface for each voxel. Assuming that we have the global position of the camera C^n, i.e. we know the global rotation R and translation $\mathbf{t}$ of the camera at time step n, we can express the coordinates of the voxel V_{ijk} in the cameras frame of reference by computing

$$\mathbf{x}_L = R^T \mathbf{x}_G - R^T \mathbf{t}, \tag{3.1}$$

where $\mathbf{x}_G$ is the known 3D coordinates for the voxel V_{ijk} in the global frame. Then, provided that $z_L > 0$ which means that the voxel is in front of the camera, we can find which pixel the voxel is projected onto by computing

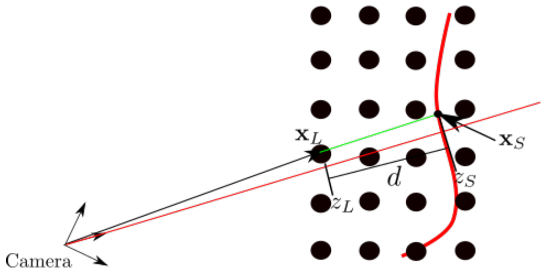


Figure 3.3: Instead of measuring the distance between $\mathbf{x}_L$ and $\mathbf{x}_S$, we measure the distance along the principal axis.

$$p_x = \frac{f x_L}{z_L} + c_x \tag{3.2}$$

$$p_y = \frac{f y_L}{z_L} + c_y. \tag{3.3}$$

The surface point observed in this pixel lies on the ray between the camera center and V_{ijk}. We can read the depth for the surface point at pixel (p_x, p_y) by

$$z = I_d(p_x, p_y). \tag{3.4}$$

With known depth z, the 3D point for the surface point $\mathbf{x}_S$ can be computed by

$$\mathbf{x}_S = \begin{pmatrix} \frac{(p_x - c_x)z}{f} \\ \frac{(p_y - c_y)z}{f} \\ z \end{pmatrix}. \tag{3.5}$$

It is now easy to compute the distance between the voxel and the surface point along the ray between the camera and the voxel via

$$d = \|\mathbf{x}_S - \mathbf{x}_L\|. \tag{3.6}$$

The sign of the distance is obtained by comparing z_L and z_S. In practice, it is easier to approximate the distance by just taking the difference between the z-coordinates, i.e.

$$d = z_L - z_S.$$

With this approximation, a voxel will have a negative distance if the voxel is in front of the surface ($z_L < z_S$) and a positive distance if it is behind. Note that this distance is an approximation of the projective distance, but in practice it does not matter. The idea is illustrated in Figure 3.3.

Since we are estimating the projected distance, it can happen at borders that a surface point close to the voxel is missed if it is not on the ray between the camera center and the voxel. Instead one might get a measurement for a surface point far away from the voxel. To reduce the impact of such erroneous measurements, the estimated distance is truncated at a positive threshold δ. We get the approximated distance

$$d_t = \begin{cases} -\delta, & d < -\delta \\ d, & |d| \leq \delta \\ \delta, & d > \delta. \end{cases} \tag{3.7}$$

This makes the potential error in the measurement limited to the bandwidth $[-\delta, \delta]$. There is also an uncertainty for the measurements when the voxel is behind the observed surface. A voxel might be close to another surface which is not observed in that frame. Therefore we also introduce a weight function for the uncertainties in the measurements. Since we cannot see behind surfaces, a lower weight is assigned to measurements behind a surface and the further behind the surface is, the lower the weight is. The weight function is defined as follows

$$w(d) = \begin{cases} 1, & d \leq \epsilon \\ e^{-\sigma(d-\epsilon)^2}, & \epsilon < d \leq \delta \\ 0, & d > \delta. \end{cases} \tag{3.8}$$

Here σ is a positive parameter and $\epsilon \leq \delta$.

The distance measurements are made for all voxels in the grid and for every frame with corresponding known global pose of the camera we get new measurements. For a voxel V_{ijk} we thus get a measurement D^n_{ijk} for each image I^n_d. The question is, how do we obtain a TSDF which takes all measurements into account? As proposed by [22], we can formulate the optimization problem

$$E(D_{ijk}) = \sum_{n=1}^{N} w^n (D_{ijk} - D^n_{ijk})^2, \tag{3.9}$$

where w^n is the weight of the measurement D^n_{ijk} and N is the number of images. Taking the derivative of this function and setting it to zero one gets

$$\sum_{n=1}^{N} w^n (D_{ijk} - D^n_{ijk}) = 0 \tag{3.10}$$

$$\Leftrightarrow$$

$$\frac{\sum_{n=1}^{N} w^n D^n_{ijk}}{\sum_{n=1}^{N} w^n} = D_{ijk}.$$

The optimal measurement for a voxel V_{ijk} is therefore the weighted average of all measurements. Since each voxel is independent of the others, one can easily obtain an optimal TSDF by computing the weighted average for a voxel independent of its neighbours. For a voxel, we do the following update

$$\begin{aligned} D^{n+1} &= \frac{W^n D^n + w(d^{n+1}) d_t^{n+1}}{W^n + w(d^{n+1})} \\ W^{n+1} &= W^n + w(d^{n+1}). \end{aligned} \tag{3.11}$$

As we can see, for a new image we can simply update the entire grid by these computations in order to get the current best approximation of the 3D model. Furthermore, each computation only needs reading and writing from one voxel, so this procedure is straightforward to implement in parallel.

Similarly, the color for voxel V_{ijk} can be estimated by extracting the RGB-vector (r, g, b) from the color image I_c^{n+1}. For each new image we obtain a measurement which we can integrate into the voxel by computing

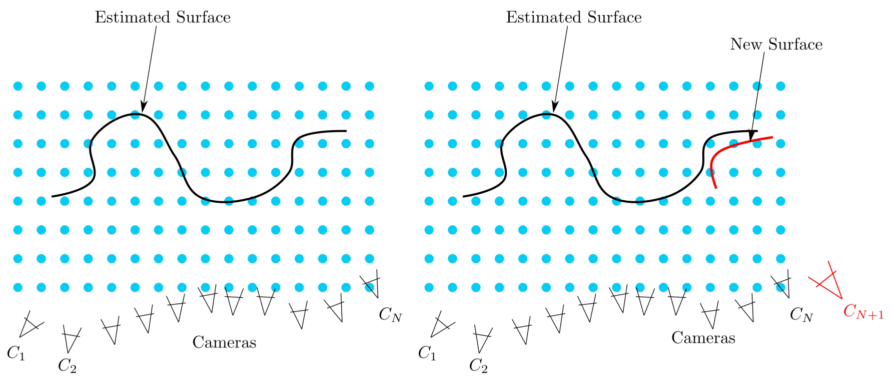
$$R^{n+1} = \frac{W_c^n R^n + w_c r^{n+1}}{W_c^n + w_c} \tag{3.12}$$

$$G^{n+1} = \frac{W_c^n G^n + w_c g^{n+1}}{W_c^n + w_c} \tag{3.13}$$

$$B^{n+1} = \frac{W_C^n B^n + w_c b^{n+1}}{W_c^n + w_c}, \tag{3.14}$$

where w_c is the weight of the new measurement defined as

$$w_c = \cos(\theta) \cdot w(d^{n+1}), \tag{3.15}$$



(a) Assuming we know the rotation and translation of the first N camera positions a surface can be created, represented in a voxel grid as a TSDF.

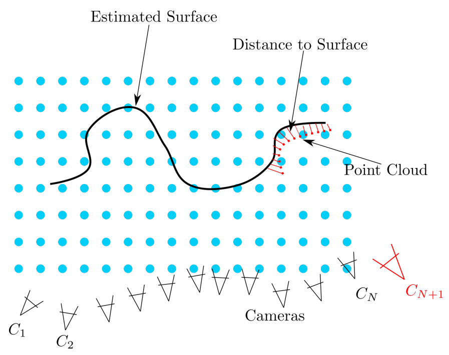
(b) Without correct rotation and translation, the newly observed surface cannot be correctly aligned to the estimated surface from the first N images.

Figure 3.4: *Left*: Illustration of how the surface might look like after N frames. *Right*: With a guess of the rotation and translation, the newly observed 3D points can be reconstructed in the grid.

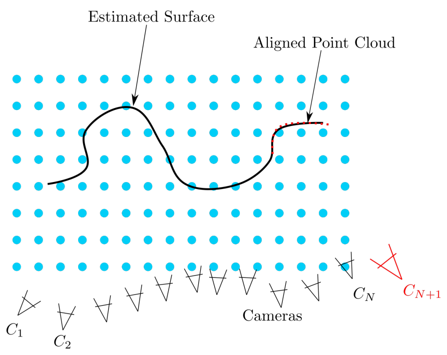
where θ is the angle between the optical axis and the light ray, $(r^{n+1}, g^{n+1}, b^{n+1})$ are the measured intensities in the new color image I_c^{n+1}. These measurements assigns an RGB-vector to every voxel. This color vector can be used to colorize the model.

3.3 Estimating the Camera Pose Using Geometry Information From the 3D Model

We now have a simple method of integrating a new depth frame into the 3D model, given that we already know the pose of the camera with respect to the global coordinate system. Simple as it may sound, it is not so easy to obtain the rotation and translation of the camera. The main contribution of this part of the thesis is how we tackle the problem of finding the camera pose using the model directly, rather than doing an ICP-like procedure or maximizing photo-consistency in a frame-to-frame manner. The key idea is to use the distance information embedded in the truncated signed distance function itself.



(a) The point cloud can be reconstructed into the voxel grid and for each point we can estimate the distance between the surface and the 3D point.



(b) With the correct rotation and translation of the camera, as many points as possible should be reconstructed onto the surface.

Figure 3.5: *Left*: Computing the distance for each reconstructed 3D point gives an error. *Right*: Minimizing the error gives a camera pose that align the point cloud to the observed model.

Assume that we after N images have found a (correct) representation of the 3D model through the TSDF, as illustrated in Figure 3.4a.

Given a new image, I_d^{N+1}, we get new measurements of the surface. Without knowing the rotation and translation of the camera, a guess will most likely not align the new surface onto the estimated (assumed true) surface, as depicted in Figure 3.4b.

The key idea to find the correct configuration of the camera is to use the distance information obtained for each 3D point from the new depth image. In reality, what we get from each new depth image is a point cloud and by a guess of the global pose of the position of the camera, one can reconstruct the point cloud into the voxel grid. For each 3D point, we can find out where in the voxel grid it is located. Using the distance information in the voxels, a distance between the 3D point and the actual surface can be computed, as shown in Figure 3.5a.

Assuming a small camera motion, most of the scene has already been observed in the previous frames. Therefore, to find the correct pose of the camera, it is reasonable to find the rotation and translation which minimizes the distances between the point cloud and the surface, as illustrated in Figure 3.5b. We now define the function for obtaining the distance to the surface.

Definition 3.3.1. Let $\phi : \mathbb{R}^3 \rightarrow \mathbb{R}$ be the function which takes a 3D point $\mathbf{x}$ and returns the value in the voxel grid at that position in the grid.

The challenge now is to find the pose. By observing that a 3D point which only lies on the surface if it has a distance of zero, the following error function can be defined

$$E(R, \mathbf{t}) = \sum_{i=1}^{M} \sum_{j=1}^{N} \phi(R\mathbf{x}_{ij} + \mathbf{t})^2. \tag{3.16}$$

Here R and $\mathbf{t}$ denotes the global rotation and translation and $\mathbf{x}_{ij}$ is the local 3D point from pixel (i, j) and ϕ is the TSDF. M and N are the number of rows and columns in the image. If this error function is 0 for some R and $\mathbf{t}$, all points are reconstructed on the surface. Due to noise and earlier unobserved structure, the error function will in practice never become zero, so we need to find the minimal error. The task is now to solve

$$\min_{R, \mathbf{t}} \sum_{i=1}^{M} \sum_{j=1}^{N} \phi(R\mathbf{x}_{ij} + \mathbf{t})^2. \tag{3.17}$$

However, we have no analytic expression of the signed distance function which makes it hard to minimize directly. To parametrize R and $\mathbf{t}$ we use the Lie Algebra representation [48]. With this representation, it is possible to represent the entire rigid transformation via a 6-dimensional vector

$$(\boldsymbol{\omega}, \mathbf{t}) = (r_x, r_y, r_z, t_x, t_y, t_z). \tag{3.18}$$

Here r_x, r_y and r_z represent the rotation around the three axes and t_x, t_y and t_z is the translation. To go from this vector representation to a rigid transformation $R \in \mathcal{SO}(3)$, one computes the exponential matrix

$$R = e^{\hat{\boldsymbol{\omega}}}, \tag{3.19}$$

where

$$\hat{\boldsymbol{\omega}} = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}. \tag{3.20}$$

```
Data: Depth image: $I_d^{N+1}$, SDF: $\phi$, Previous camera position: $(\boldsymbol{\omega}^N, \mathbf{t}^N)$
Result: Camera position $(\boldsymbol{\omega}^{N+1}, \mathbf{t}^{N+1})$
$\boldsymbol{\omega}_{New} = \boldsymbol{\omega}^N$;
$\mathbf{t}_{New} = \mathbf{t}^N$ ;
$\boldsymbol{\omega}_{Old} = \boldsymbol{\omega}^N$;
$\mathbf{t}_{Old} = \mathbf{t}^N$ ;
while not converged do
    $\boldsymbol{\omega}_{Old} = \boldsymbol{\omega}_{New}$ ;
    $\mathbf{t}_{Old} = \mathbf{t}_{New}$ ;
    $A = \sum_{i,j} (\nabla\phi(g(\boldsymbol{\omega}_{New}, \mathbf{t}_{New}, \mathbf{x}_{ij})))(\nabla\phi(g(\boldsymbol{\omega}_{New}, \mathbf{t}_{New}, \mathbf{x}_{ij})))^T$;
    $\mathbf{b} = \sum_{i,j} \phi(g(\boldsymbol{\omega}_{New}, \mathbf{t}_{New}, \mathbf{x}_{ij}))\nabla\phi(g(\boldsymbol{\omega}_{New}, \mathbf{t}_{New}, \mathbf{x}_{ij}))$;
    $\begin{pmatrix}\boldsymbol{\omega}_{New}\\ \mathbf{t}_{New}\end{pmatrix} = -A^{-1}\mathbf{b} + \begin{pmatrix}\boldsymbol{\omega}_{New}\\ \mathbf{t}_{New}\end{pmatrix}$;
    if $\|(\boldsymbol{\omega}_{New}, \mathbf{t}_{New}) - (\boldsymbol{\omega}_{Old}, \mathbf{t}_{Old})\|_\infty < \epsilon$ then
        converged = true;
    end
end
$\boldsymbol{\omega}^{N+1} = \boldsymbol{\omega}_{New}$;
$\mathbf{t}^{N+1} = \mathbf{t}_{New}$
```

Algorithm 1: The algorithm for computing the new camera pose $(\boldsymbol{\omega}^{N+1}, \mathbf{t}^{N+1})$.

With this representation, we can rewrite the error function as

$$E(\boldsymbol{\omega}, \mathbf{t}) = \sum_{i=1}^{M} \sum_{j=1}^{N} \phi(g(\boldsymbol{\omega}, \mathbf{t}, \mathbf{x}_{ij}))^2, \tag{3.21}$$

where

$$g(\boldsymbol{\omega}, \mathbf{t}, \mathbf{x}) = e^{\hat{\omega}} \mathbf{x} + \mathbf{t}. \tag{3.22}$$

In order to optimize the above energy, we use the Gauss-Newton method. The Gauss-Newton method is suitable since the distance between two consecutive camera positions will be small under normal circumstances due to the high frame-rate. Hence by using the last known position, we will be quite close to the optimal point already. Linearizing the error function around the current guess of the camera position $(\boldsymbol{\omega}_0, \mathbf{t_0})$, we get the following error function

$$E(\boldsymbol{\omega}, \mathbf{t}) \approx \sum_{i=1}^{M} \sum_{j=1}^{N} (\phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})) + (\nabla \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})))^T \begin{pmatrix} \boldsymbol{\omega} - \boldsymbol{\omega}_0 \\ \mathbf{t} - \mathbf{t}_0 \end{pmatrix})^2. \tag{3.23}$$

Now we can optimize the approximated error function by taking the gradient of it with respect to $(\boldsymbol{\omega}, \mathbf{t})$ and setting it equal to 0. We get

$$\sum_{i,j} \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})) \nabla \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})) +$$
$$(\nabla \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})))(\nabla \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})))^T \begin{pmatrix} \boldsymbol{\omega} - \boldsymbol{\omega}_0 \\ \mathbf{t} - \mathbf{t}_0 \end{pmatrix} = 0, \tag{3.24}$$

which is easily solved if the resulting matrix

$$A = \sum_{i,j} (\nabla \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})))(\nabla \phi(g(\boldsymbol{\omega}_0, \mathbf{t}_0, \mathbf{x}_{ij})))^T \tag{3.25}$$

is invertible. This will be the case as long as there is enough different structure in the scene. Then there will be a unique point $(\boldsymbol{\omega}^*, \boldsymbol{t}^*)$ such that error is minimized. In case of planar scenes, this will fail since we do not have enough constraints to determine the rotation and translation uniquely. This is a general shortcoming of purely geometry based tracking approaches, such as ICP.

To find the camera position, we initialize with the estimated camera position from the last frame. Then we start by solving the linearized error function and take the newly found camera pose and re-linearize again until convergence. After convergence we have the camera pose and can update the 3D model. The method is summarized in Algorithm 1.

Note that we are computing the gradient for each 3D point obtained from pixel (i, j) and the computations are independent of each other. Hence, these computations can mostly be done in parallel, it takes some more refined techniques to implement it compared to integrating the 3D models though. That is because at the end all matrices A_{ij} and b_{ij} computed for each pixel (i, j) need to be summed up to one matrix A and one vector b. To do that on the GPU, we have to use tree-reduction and use shared memory. Otherwise, it is quite straightforward.

The main difference between our approach and KinectFusion [50] is that we make use of all 3D points and we make no explicit data association, which is done in KinectFusion and their ICP-framework.

```
Data: Depth Image: $I_d^n$, SDF: $\phi$
\\Start solution;
$(\boldsymbol{\omega}^0, \mathbf{t}^0) = (\boldsymbol{\omega}_{init}, \mathbf{t}_{init})$;
$\phi = $ updateSDF($\phi, \boldsymbol{\omega}^0, \mathbf{t}^0, I_d^0$);
$k = 1$;
while stop == false do
    $I_d^n$ = acquireDepthImage();
    $(\boldsymbol{\omega}^n, \mathbf{t}^n)$ = getCameraPose($\boldsymbol{\omega}^{n-1}, \mathbf{t}^{n-1}, I_d^n, \phi$);
    $\phi$ = updateSDF($\phi, \boldsymbol{\omega}^n, \mathbf{t}^n, I_d^n$);
    $n++$;
    if stopping criteria fulfilled then
        stop = true;
    end
end
```

Algorithm 2: Work flow for the system to create a complete 3D model. To stop the procedure one can for instance pre-define how many images one should use. Other stopping criteria are of course possible.

In summary, we now have a way of estimating the camera pose given a 3D

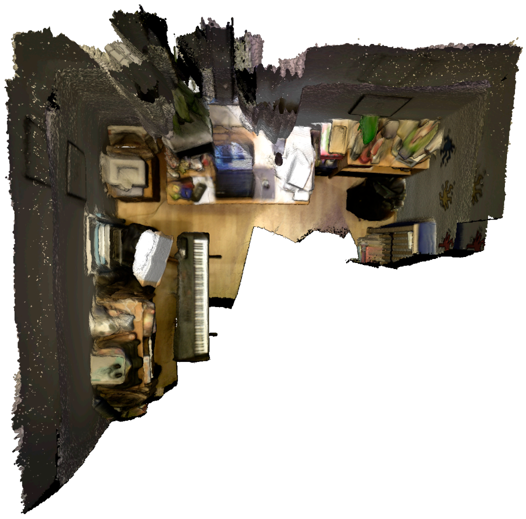


Figure 3.6: 3D reconstruction of a room, top view.

model and a new image. Moreover, we also know how to integrate this image into the model in order to update it. This can be used to create an algorithm capable of creating a 3D model on the fly. The work flow for the method is presented in Algorithm 2.

3.4 Results and Experiments

Here we evaluate our proposed method, both qualitatively and quantitatively. Several 3D models are presented, obtained from real data. We also show that a quadrocopter is capable of navigating on its own using our algorithm for localization. To evaluate our method quantitatively, it is tested on the public benchmark [66].

3.4.1 Qualitative Results

The advantage of using a TSDF is that it requires no constraints on the topology of the surface. This means that it should be possible to create arbitrary 3D models, provided there is enough structure to get the tracking to work. To demonstrate this, look at Figures 3.6 and 3.7. This is a smaller room which has been recon-

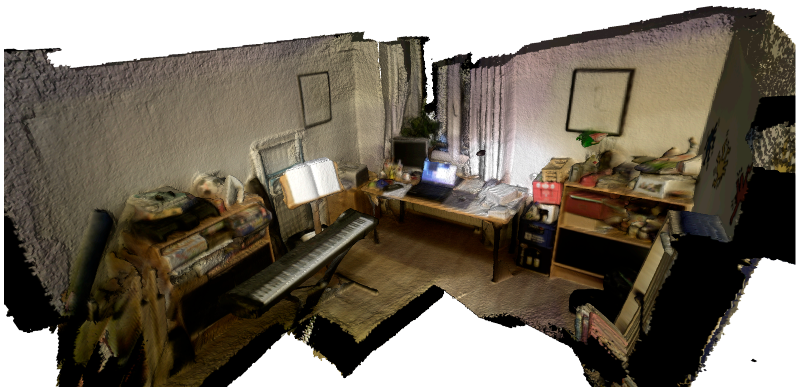


Figure 3.7: Same model as in Figure 3.6, note that the black keys on the keyboard are distinguishable from the white keys. That is an indication of a consistent tracking.

structed using the proposed method. As can be seen, it looks quite decent. For instance the black keys on the keyboard are distinguishable, which indicates a correct estimation of the trajectory. Also the figures on the wall to the right in Figure 3.7 have distinct edges and are correctly reconstructed, if the tracking had failed, they would have been smeared out. However, looking at the wall in the left corner at the window, there seems to be some artifacts and the motive on the pictures are not distinguishable. Though the picture frames are sharp rectangles and correctly reconstructed, there are still some challenges to be met.

A disadvantage with the uniform grid representation of the TSDF is that it requires a lot of memory and to make detailed reconstructions, the object cannot be too large. For instance, a grid of size 4 m^3 would have a spatial resolution of approximately 1.6 cm if 256^3 voxels are used. To get detailed reconstructions one has to use small objects which fits into a small voxel grid. That the reconstruction can be made more detailed is clearly illustrated in Figure 3.8, here the emblem on the book is clearly distinguishable for example. The high quality of the model itself is of course proof of a good estimation of the camera trajectory. To verify this we look at Figure 3.9 where we can see that there is very little drift in the sequence due to the closed loop. Note that the reconstruction could not have such high resolution if the scene were larger. Here the volume is approximately 1 m^3.

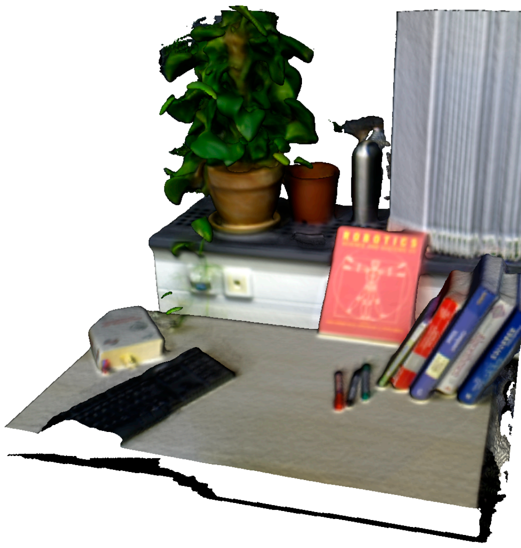


Figure 3.8: A reconstruction of a smaller object. Due to the small size of the grid, the reconstruction gets significantly more detailed. Look at the emblem on the book, it is clearly distinguishable.

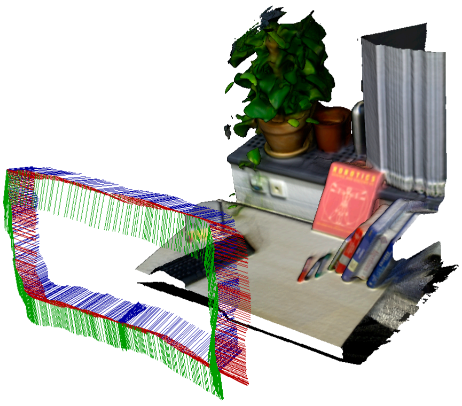


Figure 3.9: Looking at the trajectory and the quality of the 3D model, it is apparent that there is little drift in this sequence.

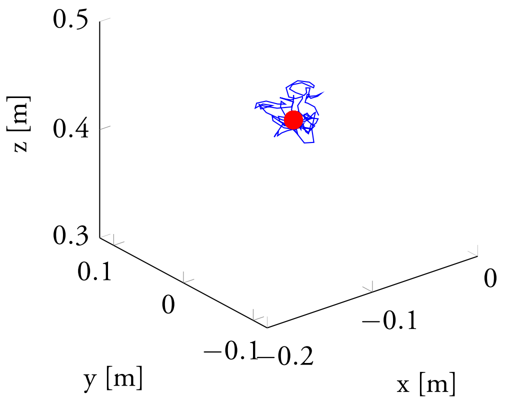


Figure 3.10: In the first quadrocopter experiment the task of the quadrocopter was to hover at the same point. As can be seen in the image, it is capable of staying at the same position reasonably well.

One application of real-time 3D reconstruction mentioned in the introduction could for instance be autonomous flights of a quadrocopter. We investigate this further by performing several experiments using a quadrocopter together with an Asus sensor and base station with GPU capabilities. The first experiments were made to see if a quadrocopter was capable of following a pre-defined orbit. The model was initialized while the quadrocopter was staying on the ground, then a signal was sent to start and lift to a certain altitude. Thereafter the mode was switched to autonomous control and the quadrocopter should follow a predefined orbit by computing its position using the depth sensor and our algorithm. The first experiment was just to hover on the same position, which resulted in an average standard deviation of 2.1 cm. This is demonstrated in Figure 3.10. The error was measured between the set goal position and the estimated position of our algorithm. A more advanced trajectory was also tested. The task was to navigate in a rectangle and follow the path for several rounds. As can be seen in Figure 3.11, this was successfully accomplished. The blue line is the pose the quadrocopter flew and the red line is the path it was supposed to take. In another experiment, the quadrocopter was in assisted mode and a user should specify way points to which the quadrocopter shoud navigate. The resulting 3D model is decent as can be seen in Figure 3.12 and Figure 3.13.

The algorithm can also be used to create 3D models of persons. By sitting

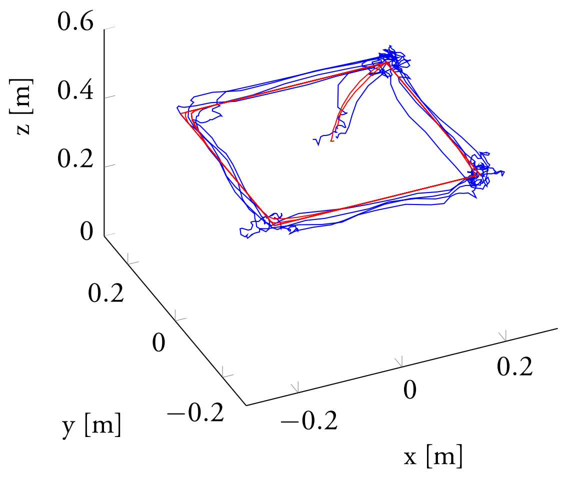


Figure 3.11: The path the quadrocopter should follow is in red and the estimated trajectory in blue. As can be seen in the image, it manages to follow the same trajectory for several rounds.

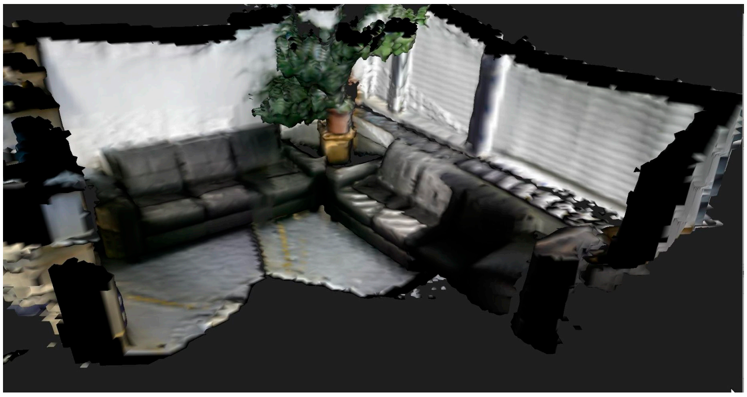


Figure 3.12: Side view of the resulting 3D reconstruction of a room using the assisted mode on the quadrocopter.

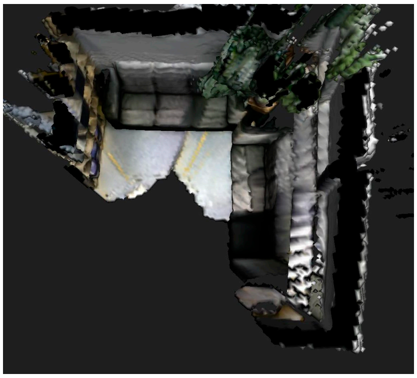


Figure 3.13: Top view of the 3D model obtained from assisted mode using the quadrocopter.

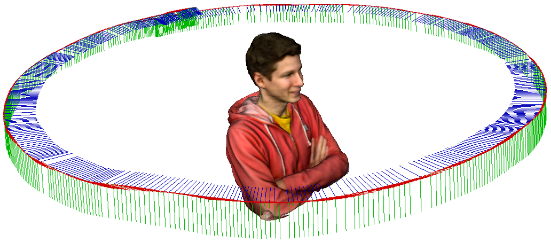


Figure 3.14: A 3D scan of a man sitting on a swivel chair. The colored lines is the estimated pose of the camera.

on a swivel chair and rotate 360° one gets a complete scan of the upper body. A result can be seen in Figure 3.14. Looking at the pose in Figure 3.14 we see that the loop closes nicely, which it is supposed to do.

3.4.2 Quantitative Results

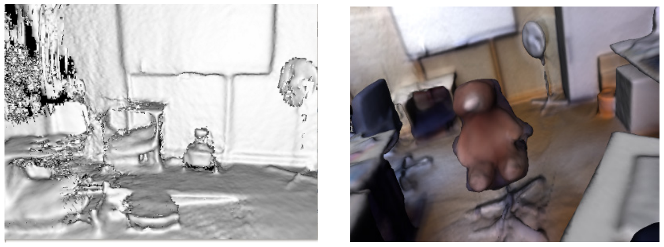
Here we evaluate our proposed method quantitatively and compare to [1], which is an open source implementation of KinectFusion [50]. We also provide results from [26] for a comparison. However, it should be noted that [26] does not solve the online-problem. That means all images are captured before the camera pose is globally optimized and the surface created. In contrast we find the camera pose as we acquire images and estimates the surface at the same time, just like

Table 3.1: The root-mean square absolute trajectory error for KinFu and our method for different resolutions, metrics and datasets. We also provide results from [26], although they do not solve the online problem.

Method	Res.	Teddy	F1 Desk	F1 Desk2	F3 Household
KinFu [1]	256	0.156 m	0.057 m	0.420 m	0.064 m
KinFu [1]	512	0.337 m	0.068 m	0.635 m	0.061 m
Point-To-Point	256	0.075 m	0.037 m	0.064 m	0.037 m
Point-To-Point	512	**0.072** m	**0.035** m	**0.055** m	**0.035** m
RGB-D SLAM [26]		0.111 m	0.026 m	0.043 m	0.059 m

Method	Res.	F1 360	F1 Plant	F1 RPY	F1 XYZ
KinFu [1]	256	0.913 m	0.598 m	0.133 m	0.026 m
KinFu [1]	512	0.591 m	0.281 m	0.081 m	0.025 m
Point-To-Point	256	0.553 m	0.048 m	**0.042** m	0.022 m
Point-To-Point	512	**0.131** m	**0.044** m	0.045 m	**0.022** m
RGB-D SLAM [26]		0.071 m	0.061 m	0.029 m	0.013 m

KinectFusion. However, it can still be of interest to see how our method performs compared to [26]. The results are provided in Table 3.1. For these experiments we used an NVidia Geforce GTX 770, Intel i7 3.4 GHz processor and 16 GB RAM. The frame rate with a voxel grid of 512^3 voxels was about $15 - 16$ Hz with this hardware. Looking at Table 3.1, we see that our method clearly outperforms KinFu on almost all datasets and we are comparable to [26]. Either KinFu works poorly, or not at all. There might be some difference between the KinFu implementation and the original KinectFusion, we tried however to tweak the parameters of [1] as good as possible. Inaccuracies in the estimation of the camera positions lead to errors in the TSDF which ultimately destroys the implicit surface, which is shown in Figure 3.15 and demonstrates that our method truly works better for this dataset. The main reason to why our method is superior to KinectFusion is probably that we make use of more information when we compute the rotation and translation of the camera. In KinectFusion, a point cloud is


(a) Reconstruction of Fr1 Teddy using KinFu

(b) Reconstruction of Fr1 Teddy using our method.

Figure 3.15: Comparison of the reconstruction of Fr1 Teddy using our method and KinFu. Note that the Teddy Bear in the left figure is completely gone.

created by performing ray casting on the current 3D model. Then the 3D points in this point cloud is associated to the points in the new point cloud obtained from the new depth image, through the fast Data Association algorithm [10]. In this process, potential matches are rejected, so at the end, KinectFusion uses less 3D points to compute the camera transformation. In contrast, our approach uses all 3D points which are reconstructed in the grid where we have measurements of the distance to the surface. This way we use more data, which probably makes tracking more robust and accurate. It might also be that we get a better estimation of the error between the surface and the reconstructed 3D point compared to KinectFusion. In the data association there might be false matches which will have a negative effect on the tracking.

To get these results, we used a threshold for the truncation of $\delta = 0.3$ m. This is a rather wide threshold, but it is empirically found that this gave the best result. In contrast, KinectFusion uses a much smaller threshold which can resolve finer details.

3.5 Conclusion

In this chapter we have seen that the distance information in the TSDF can be used to estimate the camera pose. We have also seen how to update the surface on the fly and that most of the algorithm can be parallelized which allows for a great speed-up. Finally in the experiments it has been shown that our method works well on real data and that we outperform KinFu [1] on benchmarks.

Chapter 4

Robust Estimation of the Camera Pose

4.1 Introduction

In Chapter 3 it was demonstrated how the camera pose could be estimated using the TSDF and how the model could be updated simultaneously. For applications like robotic navigation and virtual reality, it is important to be able to estimate the pose of the camera in varying environments. A drawback with geometric based tracking algorithms like ours in Chapter 3 is that the scene cannot be completely planar. If it is, there are not enough constraints to determine the rotation and translation of the camera. In this chapter, we will see that we can improve the camera tracking in several ways. The main objective is to make the pose estimation more robust to handle different environments. We will see how we can use the color information of the surface to improve the robustness and accuracy of the camera pose. It will also be shown that by combining information in the captured color images, the pose estimation can be made less sensitive to the assumption of a small camera motion. A method to reduce the memory consumption is also presented.

4.2 Invoking Color in the Camera Pose Estimation

We saw in Section 3.2 how the color of the 3D model can be estimated by computing a weighted average of color measurements for each voxel. Now when we have a textured 3D model represented in a voxel grid, we want to use this information to improve the camera tracking. The idea is based on photo consistency between the current estimation of the 3D model and the newly obtained color image I_c. By using a guess of the transformation $[R, \mathbf{t}]$, we can from pixel (i, j)

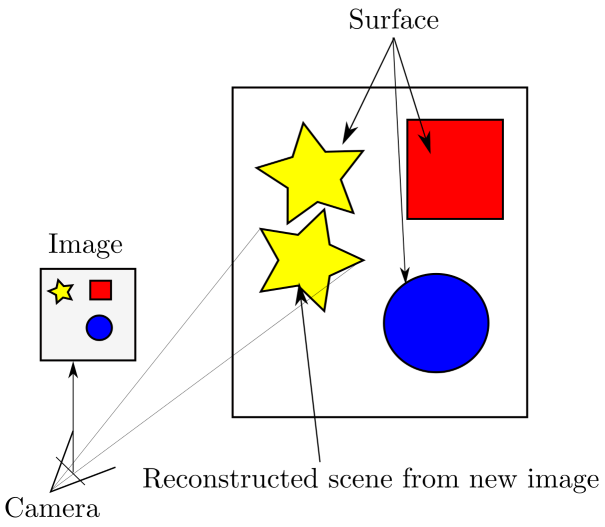


Figure 4.1: By reconstructing the 3D points with a guess of R and $\mathbf{t}$, it is possible to compare the color of the model where the point is reconstructed and the color in the color image I_c. Ideally, it should match.

reconstruct a 3D point into the voxel grid by

$$\mathbf{x}_G = R\mathbf{x}_L + \mathbf{t}, \tag{4.1}$$

where

$$\mathbf{x}_L = \begin{pmatrix} \frac{(i-c_x)z}{f} \\ \frac{(j-c_y)z}{f} \\ z \end{pmatrix}. \tag{4.2}$$

From $\mathbf{x}_G$, we can easily find out where in the voxel grid the point is located and we can then compute the color (R, G, B) of the surface there. By obtaining the RGB-vector of the model (R, G, B), we can compare this with the color intensities in the pixel (i, j) in the RGB image I_c. Ideally, these should match for each pixel, as illustrated in Figure 4.1.

By computing the difference between the intensities in the image and the color on the surface, we get the error

$$E_{color}(R, \mathbf{t}) = \sum_{ij} \|C(R\mathbf{x}_{ij} + \mathbf{t}) - I_c(i, j)\|^2, \tag{4.3}$$

where $C(R\mathbf{x}_{ij}+\mathbf{t})$ is the RGB-vector in the voxel grid at $R\mathbf{x}_{ij}+\mathbf{t}$ and $I_c(i,j)$ is the RGB-vector in the color image at pixel (i,j). Thus, we have an error metric which takes color information into account. This allows us to integrate color information into our original method where we only use geometric information. Let us first define the color error as

$$\psi(R\mathbf{x}_{ij}+\mathbf{t}) = \|C(R\mathbf{x}_{ij}+\mathbf{t}) - I_c(i,j)\|. \tag{4.4}$$

Adding this to the geometric error we get

$$E_{tot}(R,\mathbf{t}) = \phi(R\mathbf{x}_{ij}+\mathbf{t})^2 + \alpha\psi(R\mathbf{x}_{ij}+\mathbf{t})^2, \tag{4.5}$$

where α is the weight of the color error. This term takes into account both the geometric error and error in each color channel. The error is zero when the rotation and translation is such that the point is reconstructed onto the zero level set and the color on the surface matches the color in the pixel.

With this, the new error function we want to minimize is

$$E(R,\mathbf{t}) = \sum_{i,j} \phi(R\mathbf{x}_{ij}+\mathbf{t})^2 + \alpha\psi(R\mathbf{x}_{ij}+\mathbf{t})^2, \tag{4.6}$$

where we sum over all pixels (i,j). Again, we change representation of the rigid body motion by using the Lie-algebra representation

$$(\boldsymbol{\omega},\mathbf{t}) = (\omega_x,\omega_y,\omega_z,t_x,t_y,t_z). \tag{4.7}$$

With this and a local 3D coordinate $\mathbf{x}_{ij}$ for pixel (i,j) we write the residual vector as

$$r_{ij}(\boldsymbol{\omega},\mathbf{t},\mathbf{x}_{ij}) = (\phi(g(\boldsymbol{\omega},\mathbf{t},\mathbf{x}_{ij})), \sqrt{\alpha}\psi(g(\boldsymbol{\omega},\mathbf{t},\mathbf{x}_{ij})))^T, \tag{4.8}$$

and the error function becomes

$$E(\boldsymbol{\omega},\boldsymbol{t}) = \sum_{i,j} r_{ij}(\boldsymbol{\omega},\mathbf{t},\mathbf{x}_{ij})^T r_{ij}(\boldsymbol{\omega},\mathbf{t},\mathbf{x}_{ij}), \tag{4.9}$$

which we minimize using the Gauss-Newton method. Thus we now have a method for estimating the camera position by using both geometry and texture information.

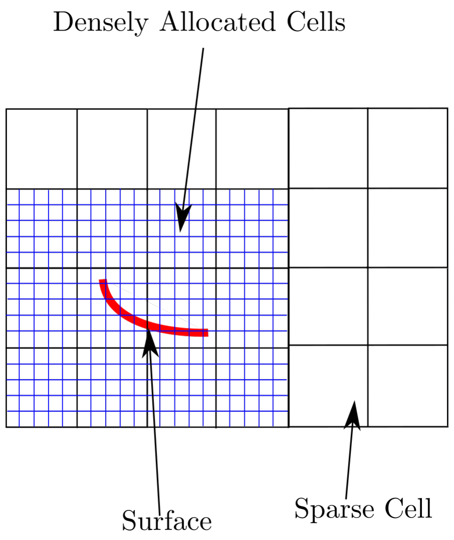


Figure 4.2: Close to the surface we allocate a dense voxel grid, but further away from the surface there are no voxels allocated.

4.2.1 Efficient Use of Memory

Another problem we address is the waste of memory by using a uniform grid. Remember that the memory consumption of a uniform grid grows cubically with the voxel resolution. More detailed reconstructions thus requires small objects, otherwise the memory consumption would explode. To address this we simply notice that we are only interested in the distance function in the vicinity of the surface and therefore there is no need to allocate a lot of voxels in empty space. The simplest approach would be to implement an octree representation. However, then the resolution quickly decreases as we get far away from the surface. To compute the derivatives, we want to have a high resolution around the surface as well. Therefore, we implement a representation where we have a very coarse voxel grid, with no voxels allocated, then if we detect a surface in any of these voxels, we allocate densely with voxels there, this is illustrated in Figure 4.2. This allows a higher spatial resolution for larger reconstructions without running out of memory.

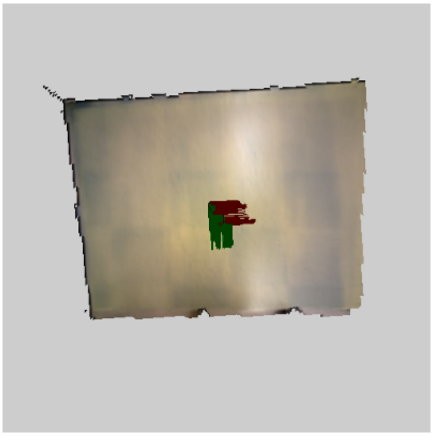


Figure 4.3: The reconstructed scene when the sensor is only facing the floor.

4.3 Experiments and Results

The purpose of invoking color information in the tracking procedure was to address that our original method would not work when there is only planar surfaces. To test our hypothesis, we made a simple experiment by recording data with the depth camera facing the floor only. As can be seen in Figure 4.3, everything looks smeared out and the estimated pose is approximately a non-moving one, whereas invoking color information yields a completely different result, as seen in Figure 4.4.

When we use the color information to estimate the camera pose, then the floor with its texture is clearly visible. Moreover, the edges on the blue squares are clearly distinguishable. The sharp edges indicate that the pose of the camera is correctly estimated. In another experiment, we tested how the method works for larger reconstructions, since we now have a more efficient way of representing the distance function. We tried to reconstruct a part of a corridor, which is quite a challenging task, since there are many scenes with little structure. The result is shown in Figure 4.5. The movement in these scenes is quite simple. Nonetheless, it shows that our tracking is also rather robust over larger scenes. The corridor is approximately 40 m long and even if there is visible drift, it is not that bad. It is inherently prone to drift since an error in the model gives an error in the tracking and then it is impossible to recover without any external technique. Nonetheless,

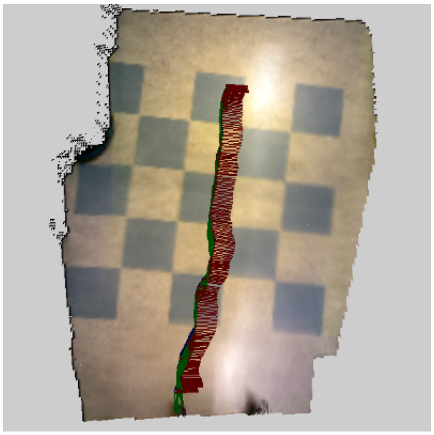


Figure 4.4: The reconstructed scene when the sensor is only facing the floor and color information is invoked in the tracking.

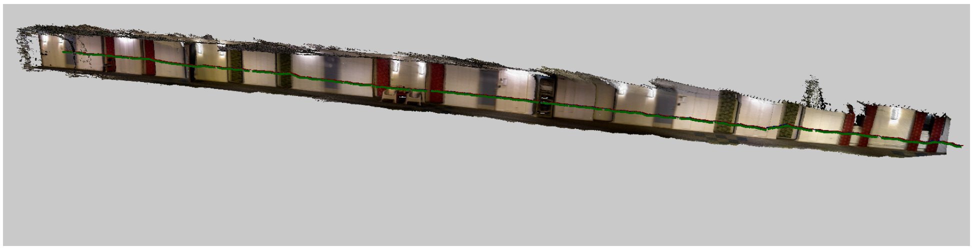


Figure 4.5: The reconstructed corridor, using both color and geometric information in the camera pose estimation.

this experiments suggests that the drift is small.

4.3.1 Quantitative Results

To evaluate our method we test it on the benchmarks [66] again. We test it on several different datasets and we also provide results from [64] for the datasets where their results are available. Just like we do, [64] includes both color- and depth-information into their tracking framework. In contrast to our work, the pose is estimated after all images are captured and the pose is globally optimized using loop closure and graph optimization [43]. It can therefore be expected that their result should be better, but it is still interesting to see the difference between online- and offline pose estimation. Several of the datasets can also be compared to KinFu [1] in Chapter 3.

It is clear that invoking color information increases the performance on most datasets. In particular, Fr3 No_Structure_Texture_Far shows the strength of invoking color information. With no color the RMSE is 1.36 m, but invoking color information decreases the error to 0.03 m. However, Fr1 Desk2 shows the opposite behvaiour, the more color that is used, the more inaccurate is the tracking. Fr1 Desk2 is considered to be a quite challenging dataset with a fast movement of the camera. It is also clear from the results that different datasets give best result for different values of the weight α. At the moment the weight has to be set manually. It would be interesting to look further into how the weight can be set automatically. The frame rate with a voxel grid of 512^3 voxels was about $11 - 12$ Hz, using an NVidia Geforce GTX 770, Intel i7 3.4 GHz processor and 16 GB RAM.

4.4 Combining Sparse and Dense Tracking

Invoking color information from the global model clearly improves the tracking as seen in Section 4.3.1. Yet, there are still situations where texture information in the model is not enough. Imagine we have a planar surface with texture, but the texture is not so distinct. The estimated model is then likely to be smeared out and it will be difficult to estimate the pose.

An idea to attack this problem is to use sparse feature points and invoke these into our error function. The approach is to find corresponding points between the new image and a sequence of images already obtained. With corresponding pixel pairs $[(p_x^{n+1}, p_y^{n+1}), (p_x^n, p_y^n)]$, where (p_x^{n+1}, p_y^{n+1}) corresponds to a pixel

Table 4.1: The root-mean square absolute trajectory error (m) for different values of the weight α. Note that $\alpha = 0$ corresponds to the pure geometric tracking approach in Section 3.3

Dataset	Weight α					
α	0	0.1	0.2	0.3	0.4	0.5
fr1 teddy	0.072	0.067	0.064	0.061	0.059	**0.058**
fr3 str_no_tx_far	**0.034**	**0.034**	0.035	0.036	0.036	0.037
fr3 no_str_tx_far	1.36	0.042	0.041	0.031	0.031	0.030
fr1 desk	**0.035**	0.036	0.036	0.037	0.037	0.038
fr1 desk2	**0.055**	0.064	0.079	0.105	0.117	0.130
fr1 360	0.131	0.131	**0.129**	0.147	0.138	0.200
fr3 office_hhould	0.035	0.032	0.027	0.025	**0.024**	**0.024**
fr1 plant	0.044	**0.042**	0.044	0.047	0.048	0.050
fr1 rpy	0.045	0.041	0.039	0.038	**0.037**	**0.037**

α	0.6	0.7	0.8	0.9	1.0	[64]
fr1 teddy	0.066	0.072	0.080	0.248	0.223	*0.036*
fr3 str_no_text_far	0.037	0.038	0.038	0.039	0.040	
fr3 no_str_text_far	0.030	0.030	0.030	**0.029**	**0.029**	
fr1 desk	0.038	0.038	0.038	0.039	0.039	*0.021*
fr1 desk2	0.132	0.135	0.138	0.140	0.140	*0.027*
fr1 360	0.196	0.199	0.205	0.737	0.759	*0.073*
fr3 office_household	**0.024**	**0.024**	0.025	0.025	0.026	*0.030*
fr1 plant	0.051	0.051	0.051	0.051	0.052	
fr1 rpy	**0.037**	**0.037**	**0.037**	**0.037**	**0.037**	

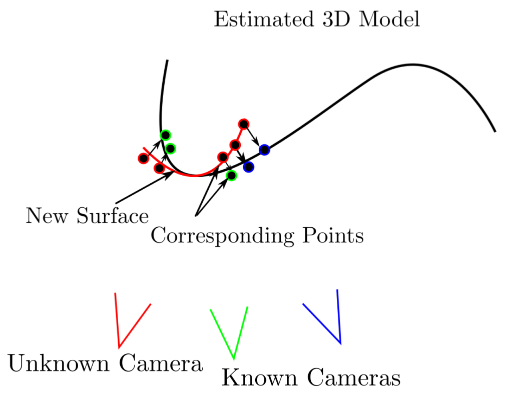


Figure 4.6: By finding feature point correspondences between the new image and an older images, one want to align these matches as good as possible.

in image I_c^{n+1} and (p_x^n, p_y^n) to a pixel in image I_c^n, we can compute the global position $\mathbf{y}_G$ of the point corresponding to (p_x^n, p_y^n) since the camera position for that frame is known. With these global 3D coordinates of $\mathbf{y}_G$, we want to find R and $\mathbf{t}$ such that

$$R\mathbf{x} + \mathbf{t} = \mathbf{y}_G. \tag{4.10}$$

where $\mathbf{x}$ is the corresponding local point in the new camera's frame of reference. The idea is depicted in Figure 4.6. By matching feature points between a sequence of the $K + 1$ latest image pairs

$$(I_c^{n+1}, I_c^n), (I_c^{n+1}, I_c^{n-1}), ..., (I_c^{n+1}, I_c^{n-K}) \tag{4.11}$$

we get a set

$$(Y_1, Y_2, ..., Y_K) \tag{4.12}$$

of global 3D points and a corresponding set of local 3D points

$$(X_1, X_2, ..., X_K). \tag{4.13}$$

Each Y_i and corresponding X_i consists of M_i number of corresponding 3D points

$$X_i = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{M_i}\} \tag{4.14}$$

$$Y_i = \{\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_{M_i}\}. \tag{4.15}$$

We can use this to define the error function

$$E(R, \mathbf{t}) = \sum_{i=1}^{K} \sum_{\substack{\mathbf{x} \in X_i \\ \mathbf{y} \in Y_i}} \|R\mathbf{x} + \mathbf{t} - \mathbf{y}\|^2 \tag{4.16}$$

where the sum is over all found corresponding 3D points.

This error function is not dependent on what the model looks like so the idea is that this term shall help finding the correct pose even if the model does not provide enough information. Furthermore, the feature based error does not require the cameras to be close to each other. Therefore, one could expect that adding this error term would make the algorithm more robust to larger camera movements. By adding (4.16) to (4.6) we get

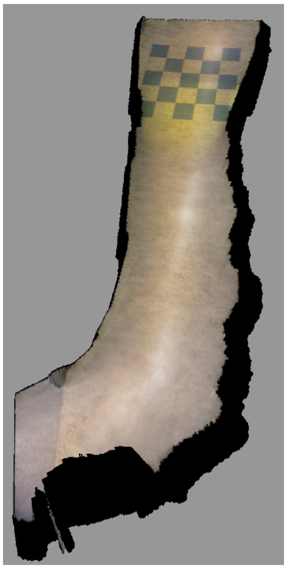
$$E(R, \mathbf{t}) = \sum_{i,j} \phi(R\mathbf{x}_{ij} + \mathbf{t})^2 + \alpha\psi(R\mathbf{x}_{ij} + \mathbf{t})^2 + \mu \sum_{l=1}^{L} \|R\mathbf{x}_l + \mathbf{t} - \mathbf{y}_l\|^2, \tag{4.17}$$

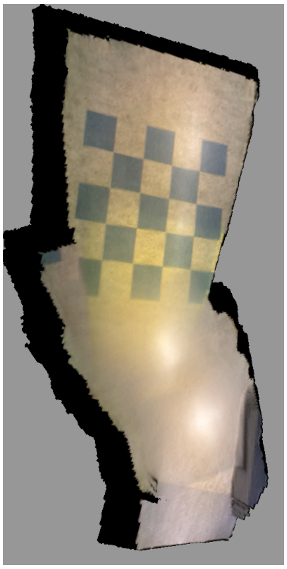
where we sum over all pixels (i, j) and all found corresponding feature points, α and μ are the weights and L is the total number of found correspondences.

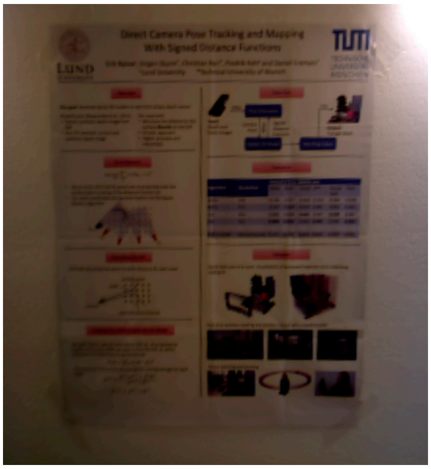
Linearizing each sum separately and then adding the resulting matrices for the optimization, we can use the Gauss-Newton method as earlier to minimize the error function.

4.4.1 Qualitative Results

This new approach was tested qualitatively on several challenging recordings. In the first sequence images from a floor with hardly any texture was recorded. This is very challenging due to the fact that it is completely planar and has little texture, so there is very little information to work with. Therefore, a purely geometric based tracker would never work and even photo consistency is hard since there is little texture and that is likely to be smeared out in the model. By also invoking sparse feature points into our tracker, the hypothesis is that this shall give more information. Looking on Figures 4.7a and 4.7b, it is clear that the extra information we get from the feature points are very helpful in these extreme situations. In the scene the recording starts in the lower left corner and ends after the chess board pattern in Figure 4.7a. When only using the model in the TSDF for tracking, the result is as in Figure 4.7b. The trajectory cannot be correctly estimated


(a) Reconstructed floor using both the model and sparse feature points


(b) Reconstructed floor using the model only.


(c) Reconstructed poster with plenty of texture using both feature points and the model.

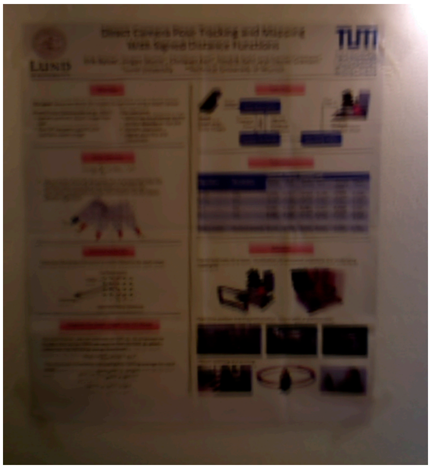
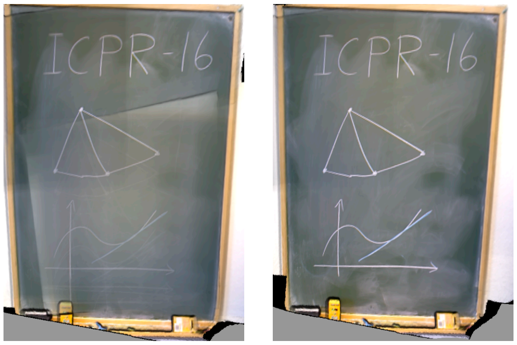

(d) Reconstructed poster using only feature points.

Figure 4.7: *Top*: Comparison between using only information in the model and also invoking feature points. *Bottom*: Comparison between using only feature points and information from both the model and feature points.



(a) Without using feature points there is not enough texture to recover the trajectory, one sees clearly that drift is present.

(b) Including feature points as well gives more accurate result for this sequence.

Figure 4.8: In this sequence one can see drift in the left figure, for example the yellow box in the lower left corner is duplicated. In contrast the right reconstruction is sharper, indicating that the pose is better estimated.

between the starting point and the blue pattern, whereas invoking the sparse feature points in the tracking gives satisfying results. We use in this and the following experiments SURF [7] to find the feature points.

One can ask oneself, do we really need the texture information in the model now? The feature points give constraints when we have planar structures. By recording a poster on a wall with plenty of texture, we used our proposed method with both geometry, texture and feature points found using SURF [7] and compared it to the obtained reconstruction where we only used sparse feature points. In Figure 4.7d, only SURF points were used and the reconstruction looks rather smeared out due to drift in the tracking. In contrast, the reconstruction using the information from the model in the TSDF as well, clearly gives a better reconstruction, as can be seen in Figure 4.7c. This can be seen by looking at the poster, the distinct details indicate that the trajectory is better estimated. Clearly, the information in the model in the TSDF reduces drift which shows that we cannot exclude texture information from the model in the tracking. Instead one should take all information into account to obtain a tracking algorithm which can work under as many circumstances as possible. For these two experiments we used $\alpha = 0.4$ and $\mu = 0.75$.

In a third experiment a recording of a blackboard was made, where there is little and not so distinct texture. The scene is also almost completely planar which makes it quite challenging. The result is shown in Figure 4.8. As can be seen, there is visual drift in Figure 4.8a where only surface- and color information in the model has been used for pose estimation. For example, the drawn graph is duplicated and the yellow box in the left corner is also duplicated. In contrast, invoking feature points in the pose estimation gives more satisfying results as seen in Figure 4.8b. The reconstruction is more distinct and the yellow box is for example better in the right image compared to the left. This is an example of where just using information form the estimated surface is not enough to give an accurate estimation of the pose. To get the results for the blackboard we used $\alpha = 0.4$ and $\mu = 3.0$.

4.4.2 Quantitative Evaluation

To measure the effect of this new approach with sparse feature points, we benchmarked it on [66]. The results are shown in Table 4.2. The parameter α in (4.17) was set to $\alpha = 0.4$. Then we let μ vary 0 and 1. As can be seen in the evaluation, adding the feature points to the camera pose estimation does not improve the accuracy for most datasets. The only noticeable difference was in datasets Desk2 and 360. Both these datasets are quite challenging and the camera movement faster than in other sequences. However, for most sequences it gives little improvement on the benchmark. The advantage of invoking the feature points in the camera tracking lies in the improved robustness as seen in Section 4.4.1. The frame rate with a uniform voxel grid of 512^3 voxels was about 4-5 Hz, using an NVidia Geforce GTX 770, Intel i7 3.4 GHz processor and 16 GB RAM.

In Table 4.3 we try to simulate a faster camera movement by using every k-th frame in order to test the robustness when the distance between the cameras increases. Many algorithms like [50], [64] and [13] rely on a small camera movement between two consecutive frames. In Table 4.3 it can be seen that our proposed method clearly gets a lower RMSE compared to our previous methods when the simulated speed of the camera is increased.

Table 4.2: The root-mean square absolute trajectory error (m) for different values of the weight μ, α was set to 0.4.

Dataset	Weight μ				
	0	0.1	0.2	0.3	0.4
fr1 teddy	**0.059**	**0.059**	0.060	0.060	0.060
fr3 str_no_txt_far	**0.036**	**0.036**	**0.036**	**0.036**	**0.036**
fr3 no_str_txt_far	0.031	**0.030**	**0.030**	**0.030**	**0.030**
fr1 desk	**0.037**	**0.037**	**0.037**	**0.037**	**0.037**
fr1 desk2	0.117	0.107	0.087	0.077	0.075
fr1 360	0.138	0.130	0.128	0.129	0.128
fr3 office_house	0.024	0.024	0.024	0.024	**0.023**
fr1 plant	0.048	0.048	0.047	0.047	0.047
fr1 rpy	0.037	0.037	0.036	0.035	0.035

Dataset	Weight μ					
	0.5	0.6	0.7	0.8	0.9	1.0
fr1 teddy	0.061	0.061	0.061	0.060	0.060	0.060
fr3 str_no_txt_far	**0.036**	0.037	0.037	0.037	0.037	0.037
fr3 no_str_txt_far	**0.030**	**0.030**	**0.030**	**0.030**	**0.030**	**0.030**
fr1 desk	**0.037**	**0.037**	**0.037**	**0.037**	**0.037**	**0.037**
fr1 desk2	0.071	0.069	0.066	**0.065**	0.066	**0.065**
fr1 360	0.121	0.118	0.117	0.115	0.112	**0.110**
fr3 office_house	**0.023**	**0.023**	**0.023**	**0.023**	**0.023**	**0.023**
fr1 plant	0.046	0.046	0.046	0.045	0.045	**0.044**
fr1 rpy	0.035	0.034	0.034	0.034	**0.033**	**0.033**

Step Size k / Dataset	1	2	3	4	5	6
Teddy [14]	**0.059**	0.473	0.656	1.142	1.098	0.990
Teddy Our	0.060	**0.059**	**0.067**	**0.065**	**0.274**	**0.268**
Desk2 [14]	0.117	0.302	**0.442**	0.820	0.715	0.991
Desk2 Our	**0.058**	**0.083**	0.526	**0.317**	**0.289**	**0.507**
360 [14]	0.131	0.288	0.773	1.419	1.774	1.939
360 Our	**0.102**	**0.010**	**0.206**	**0.581**	**1.493**	**1.254**
Plant [14]	**0.045**	0.069	0.290	0.351	0.448	**0.531**
Plant Our	0.047	**0.046**	**0.215**	**0.348**	**0.111**	0.548
Desk [14]	**0.032**	0.051	0.094	0.244	0.422	0.603
Desk Our	**0.032**	**0.033**	**0.037**	**0.045**	**0.100**	**0.320**

Table 4.3: Results on the benchmarks from [66]. We use every k-th image and compute the RMSE (m) to test the robustness for bigger distances between two consecutive frames.

4.5 Conclusion

In this part of the thesis we have seen how a 3D model represented as a TSDF can be used for estimation of the camera pose. Evaluations on benchmark show that our method gives good results on many different datasets with a reasonable speed. The more information we use, the slower the tracking is, but the more robust it becomes. Several challenges remain, for example for larger reconstructions, one must find a way of reducing errors in the model and the tracking. Now a badly estimated camera pose gets integrated into the model and that way errors are accumulated. For online reconstructions, this must be done in real-time, so that the model is correctly updated as new images are obtained. Another interesting problem is how to set the weights α and μ optimally.

Chapter 5

Low-Rank Approximation of Matrices

5.1 Introduction

Another problem studied in this thesis is how to obtain a low-rank approximation of a measurement matrix. For the reader who is not familiar with this subject we start by giving some examples of problems that eventually lead to a low-rank approximation problem.

5.1.1 Structure from Motion

One of the more familiar applications of low-rank approximation, or more specifically in this case, matrix factorization, is the work by [70]. Assume we have a set of 3D points $Q = \{\mathbf{y}_1, ..., \mathbf{y}_N\}$ where each $\mathbf{y}_i \in \mathbb{R}^3$ and K affine cameras

$$P^k = \begin{pmatrix} A^k & \mathbf{t^k} \\ \mathbf{0} & 1 \end{pmatrix}, \tag{5.1}$$

$A^k \in \mathbb{R}^{2\times 3}$ and $\mathbf{t}^k \in \mathbb{R}^{2\times 1}$. A 3D point $\mathbf{y}$ is projected onto the image plane with pixel coordinates

$$\begin{pmatrix} p_x^k \\ p_y^k \\ 1 \end{pmatrix} = P^k \begin{pmatrix} \mathbf{y} \\ 1 \end{pmatrix}. \tag{5.2}$$

Because of the special shape of the affine camera matrix the third coordinate of the projection will be one as long as the point that we are projecting is not at infinity. Therefore we can ignore this coordinate if we assume that we are only dealing with regular points.

Assume now that we have taken a sequence of K images of the same object and have tracked the image coordinates $(p^k_{x_i}, p^k_{y_i})^T$ for each 3D point $\mathbf{y}_i$ and each image I^k. We can stack all pixel coordinates in the matrix M. The rows $2k-1$ and $2k$ in M contain the coordinates for all pixels in frame I^k and M is of size $2K \times N$,

$$M = \begin{pmatrix} p^1_{x_1} & \cdots & p^1_{x_N} \\ p^1_{y_1} & \cdots & p^1_{y_N} \\ \vdots & \vdots & \vdots \\ p^K_{x_1} & \cdots & p^K_{x_N} \\ p^K_{y_1} & \cdots & p^K_{y_N} \end{pmatrix}. \tag{5.3}$$

Now we want to find the positions of the cameras and the 3D coordinates of the tracked points. Starting by finding the translation for each camera, it can be shown that

$$\mathbf{t}^k = \begin{pmatrix} \bar{p}^k_x \\ \bar{p}^k_y \end{pmatrix} - A^k \bar{\mathbf{y}}, \tag{5.4}$$

where $(\bar{p}^k_x, \bar{p}^k_y)^T$ is the mean of the observed pixels in frame k and $\bar{\mathbf{y}}$ is the mean of the 3D points. By subtracting the corresponding mean from each pixel, we can assume that the translation is zero. Thus each projection $(p^k_{x_i}, p^k_{y_i})$ in frame k can be written as

$$\begin{pmatrix} p^k_{x_i} \\ p^k_{y_i} \end{pmatrix} = A^k \mathbf{y}_i. \tag{5.5}$$

We want to factorize $M \in \mathbb{R}^{2K \times N}$ into two matrices A and Y such that

$$M = \begin{pmatrix} p^1_{x_1} & \cdots & p^1_{x_N} \\ p^1_{y_1} & \cdots & p^1_{y_N} \\ \vdots & \vdots & \vdots \\ p^K_{x_1} & \cdots & p^K_{x_N} \\ p^K_{y_1} & \cdots & p^K_{y_N} \end{pmatrix} = \begin{pmatrix} A^1 \\ \vdots \\ A^K \end{pmatrix} \begin{pmatrix} \mathbf{y}_1 & \ldots & \mathbf{y}_N \end{pmatrix} = AY. \tag{5.6}$$

Hence given a set of measurements in a matrix M the goal is to find a decomposition of M where one part corresponds to the cameras and the other corresponds to the 3D points. We can immediately conclude that the rank of the measurement matrix M is at most 3, since $A \in \mathbb{R}^{2K \times 3}$ and $Y \in \mathbb{R}^{3 \times N}$.

Now this is an ideal case, in practice we never have perfect measurements, instead there will be noise as well. The best we can do then is to find an approximation of the original data which is close to the measurements we have obtained in some norm and also fulfills certain constraints. In this case we want to optimize

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|X - M\|_F^2 \\ \text{subject to} \quad & \text{rank}(X) = 3. \end{aligned} \tag{5.7}$$

In other words we have a problem where the goal is to find a low-rank approximation of a matrix. To solve this particular problem we use von Neumanns trace theorem

$$|\text{tr}(X^T M)| \leq \sum_{i=1}^{n} \sigma_i(X)\sigma_i(M), \tag{5.8}$$

with equality when $X = U\Sigma_X V^T$ and $M = U\Sigma_M V^T$, where U and V are unitary and Σ_X and Σ_Y are diagonal matrices with the singular values of X and M respectively. It is also assumed that the singular values are assumed to be in a decreasing order. That equality holds is seen by using the properties of the scalar product for matrices

$$\langle X, M\rangle = \text{tr}(X^T M) = \text{tr}(V\Sigma_X^T U^T U \Sigma_M V^T) = \tag{5.9}$$

$$\text{tr}(V^T V \Sigma_X^T \Sigma_M) = \langle \Sigma_X, \Sigma_M \rangle. \tag{5.10}$$

Now back to (5.7),

$$\|X - M\|_F^2 = \|M\|_F^2 - 2\langle X, M\rangle + \|M\|_F^2 = \tag{5.11}$$

$$\sum_{i=1}^{n} \sigma_i(M)^2 - 2\langle X, M\rangle + \sum_{i=1}^{n} \sigma_i(X)^2 \geq \tag{5.12}$$

$$\sum_{i=1}^{n} (\sigma_i(M)^2 - 2\sigma_i(M)\sigma_i(X) + \sigma_i(X)^2) = \tag{5.13}$$

$$\sum_{i=1}^{n} (\sigma_i(X) - \sigma_i(M))^2, \tag{5.14}$$

where $\sigma_i(X)$ and $\sigma_i(M)$ are the singular values of X and Y respectively. To make $\|X - M\|_F^2$ as small as possible we see that we can choose $X = U_M \Sigma_X V_M^T$ where

U_M and V_M are unitary and obtained from SVD of M. From (5.14) we conclude that we must choose the three largest singular values of M and put them into Σ_X and set the rest of the diagonal to zero. This is because of the constraint that we want a solution of rank 3. The decomposition A and Y we are seeking can be obtained by extracting the 3 largest singular values from Σ_M and corresponding right and left singular vectors from U_M and V_M and putting them into U', Σ' and V', where $U' \in \mathbb{R}^{2K\times 3}$, $\Sigma' \in \mathbb{R}^{3\times 3}$ and $V' \in \mathbb{R}^{3\times N}$. Then we can define

$$A = U'(\Sigma')^{1/2} \tag{5.15}$$

$$Y = (\Sigma')^{1/2}V' \tag{5.16}$$

to be the camera matrices and 3D points. This is a well-known result from [70] and is an example of how low-rank approximations can be of use in computer vision. This approach was extended in [12] to the non-rigid setting, again using an affine camera.

There are several other applications of low-rank factorizations as well. For example it can also be used in Optical Flow [30], Photometric Stereo [39, 62, 6] and Non-rigid Shape Recovery [74].

5.1.2 Related Work

Typically, measurements are noisy and good optimization criterion consists of a trade-off between the rank and the residual errors, leading to a formulation of the type

$$\min_X \mu \operatorname{rank}(X) + \|X - M\|_F^2, \tag{5.17}$$

where M is a matrix of measurements.

This can be solved using SVD but it will only work if there is no missing data and it is sensitive to outliers. To handle outliers the more robust l_1 norm has been studied in [27, 65, 75].

Now assume that we have a set of images and through these images we have tracked a number of feature points. Most likely, we will not see all the points in all images throughout the whole sequence. If we put all coordinates in two rows and then stack them on top of each other for each image, we will get a matrix looking like the one in Figure 5.1.

To reconstruct the scene, we want to estimate the positions of the points in the images where they are not seen. A way of solving this problem would be to

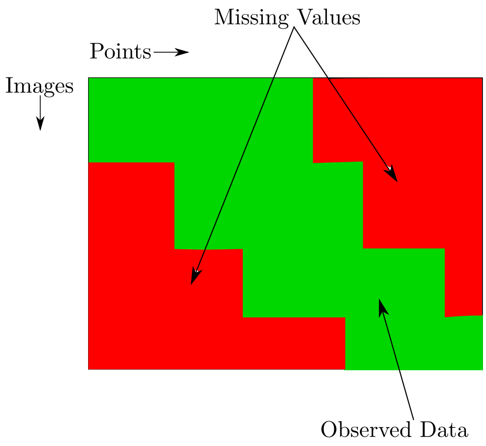


Figure 5.1: When tracking data points for example, the points will not be seen in all images. The green pattern is observed data and the red area is where the points have not been seen. In this illustration the points to the left are seen in the beginning of the sequence and the points to the right are seen in the end of the sequence.

minimize an objective of the form

$$\min_{X} \mu \operatorname{rank}(X) + \|W \odot (X - M)\|_F^2, \tag{5.18}$$

where $W_{ij} = 0$ if M_{ij} is missing and 1 otherwise and $\odot$ denotes element wise multiplication. Typically, $W \odot M$ has a high rank. The first term in (5.18) favors solutions with low rank.

Objectives of the form in (5.18) have been studied by [28, 58, 18, 51, 4] where the rank function is replaced with the nuclear norm, which is the convex envelope of the rank function on the set $\{X \in \mathbb{R}^{m\times n} | \sigma_{max}(X) \leq 1\}$. This results in the formulation

$$\min_{X} \quad \mu\|X\|_* + \|W \odot (X - M)\|_F^2, \tag{5.19}$$

where $\|X\|_*$ is the nuclear norm. In [58, 18] it is shown that this will yield an optimal solution if the location of the missing entries are random. In Structure from Motion though, the data is often highly correlated and patterns as in Figure 5.1 are common. For example in SfM it appears because tracked points typically occur in a consecutive sequence of images and then the points go out of view of the camera. Therefore, the nuclear norm might not be the best solution for some problems in computer vision. Moreover, we do not want to restrict ourselves to the set $\{X|\sigma_{max}(X) \leq 1\}$, but instead solving the problem over all matrices.

We seek to derive the convex envelope of (5.17). The advantage with working with this objective is that we translate the feasible region to lie around the measurement M. Compared to the nuclear norm which is centered around 0. The work most similar to ours is perhaps [38] which derives the convex envelope of a vector version, centered at 0.

5.2 Developing the Convex Envelope

To find a solution of (5.18), we derive the convex envelope of

$$f(X) = \mu \operatorname{rank}(X) + \|X - M\|_F^2. \tag{5.20}$$

The tactic to find a solution of (5.18) is to divide the measurement matrix M into sub-blocks with no missing data. On these sub-blocks we will be able to use our convex envelope of (5.20).

In this part we show how we can find the envelope of (5.20). We start by computing the Fenchel conjugate. Thereafter we compute the conjugate of the conjugate, which gives us the bi-conjugate which is the convex envelope of f.

The Fenchel conjugate is defined as

$$f^*(Y) = \max_X \langle X, Y\rangle - f(X). \tag{5.21}$$

Writing this as

$$f^*(Y) = \max_X \langle X, Y\rangle - (\mu \operatorname{rank}(X) + \|X - M\|_F^2), \tag{5.22}$$

we can complete squares to obtain

$$\langle X, Y\rangle - \|X\|_F^2 + 2\langle X, M\rangle - \|M\|_F^2 = \tag{5.23}$$

$$\langle X, Y + 2M\rangle - \|X\|_F^2 - \|M\|_F^2 = \tag{5.24}$$

$$-(\|X\|_F^2 - \langle X, Y + 2M\rangle) - \|M\|_F^2 = \tag{5.25}$$

$$-\|X - (\frac{1}{2}Y + M)\|_F^2 + \|\frac{1}{2}Y + M\|_F^2 - \|M\|_F^2. \tag{5.26}$$

The matrix X is of the same size as M, $m \times n$, so the rank cannot be higher than $\min(m, n)$, so by maximizing for each rank k and over all k, the Fenchel conjugate can be written as

$$\max_{k} \max_{\substack{X \\ \mathrm{rank}(X)=k}} -\mu k - \|X - (\frac{1}{2}Y + M)\|_F^2 + \|\frac{1}{2}Y + M\|_F^2 - \|M\|_F^2. \tag{5.27}$$

Here two terms are independent of X and in the inner maximization the rank k is fixed. For fixed k we can maximize with respect to X by computing the SVD of $\frac{1}{2}Y + M = U\Sigma V^T$ and setting $X = U\Sigma_k V^T$, where Σ_k only contains the k largest singular values. Inserting this into (5.27) gives

$$f^*(Y) = \max_{k} \|\frac{1}{2}Y + M\|_F^2 - \|M\|_F^2 - \sum_{i=k+1}^{n} \sigma_i^2(\frac{1}{2}Y + M) - \mu k \tag{5.28}$$

$$= \max_{k} \|\frac{1}{2}Y + M\|_F^2 - \|M\|_F^2 - \sum_{i=k+1}^{n} \sigma_i^2(\frac{1}{2}Y + M) - \sum_{i=1}^{k} \mu. \tag{5.29}$$

Looking at equation (5.29), we can deduce that the optimal k must be chosen such that

$$\sigma_k^2(\frac{1}{2}Y + M) \geq \mu \geq \sigma_{k+1}^2(\frac{1}{2}Y + M). \tag{5.30}$$

Using this we can write the conjugate function as

$$f^*(Y) = \|\frac{1}{2}Y + M\|_F^2 - \|M\|_F^2 - \sum_{i=1}^{n} \min(\mu, \sigma_i^2(\frac{1}{2}Y + M)). \tag{5.31}$$

The next step to find the convex envelope of the original function (5.20) is to compute the bi-conjugate. The bi-conjugate is defined as

$$\begin{aligned} f_X^{**} &= \max_{Y} \langle X, Y \rangle - f^*(Y) \\ &= \max_{Y} \langle X, Y \rangle - (\|\frac{1}{2}Y + M\|_F^2 - \|M\|_F^2 - \sum_{i=1}^{n} \min(\mu, \sigma_i^2(\frac{1}{2}Y + M))). \end{aligned} \tag{5.32}$$

With the change of variables

$$Z = \frac{1}{2}Y + M, \tag{5.33}$$

this becomes

$$f^{**}(X) = \max_{Z} 2\langle X, Z - M\rangle - \|Z\|_F^2 + \|M\|_F^2 + \sum_{i=1}^{n} \min(\mu, \sigma_i^2(Z)). \tag{5.34}$$

By completing squares, this can be written as

$$f^{**}(X) = \max_{Z} \|X - M\|_F^2 - \|Z - X\|_F^2 + \sum_{i=1}^{n} \min(\mu, \sigma_i^2(Z)). \tag{5.35}$$

The term $\|X - M\|_F^2$ is independent of Z, so we can write it outside the maximization

$$f^{**}(X) = \|X - M\|_F^2 + \max_{Z}(\sum_{i=1}^{n} \min(\mu, \sigma_i^2(Z)) - \|Z - X\|_F^2). \tag{5.36}$$

We also have that $-\|Z - X\|_F^2 = -\|Z\|_F^2 + 2\langle X, Z\rangle - \|X\|_F^2$ and by von Neumann's trace theorem $\langle X, Z\rangle \leq \sum_{i=1}^{n} \sigma_i(X)\sigma_i(Z)$ we get that

$$\begin{aligned} -\|Z - X\|_F^2 &= -\|Z\|_F^2 + 2\langle Z, X\rangle - \|X\|_F^2 && (5.37)\\ &= -\sum_{i=1}^{n}(\sigma_i^2(Z) + \sigma_i^2(X)) + 2\langle Z, X\rangle && (5.38)\\ &\leq \sum_{i=1}^{n} -\sigma_i^2(Z) - \sigma_i^2(X) + 2\sigma_i(Z)\sigma_i(X). && (5.39)\end{aligned}$$

To maximize (5.39), Z should have the same unitary matrices U and V as X have. The problem is now reduced to

$$\max_{Z} \sum_{i=1}^{n} \min(\mu, \sigma_i^2(Z)) - (\sigma_i(Z) - \sigma_i(X))^2. \tag{5.40}$$

To find the optimal singular values, we can maximize each term individually. Therefore,

$$\max_{\sigma_i(Z)} \min(\mu, \sigma_i^2(Z)) - (\sigma_i(Z) - \sigma_i(X))^2 \tag{5.41}$$

should be solved. There are two cases:

(i) $\sigma_i^2(Z) \leq \mu$ which gives

$$\max_{\sigma_i(Z)} 2\sigma_i(X)\sigma_i(Z) - \sigma_i(X)^2, \tag{5.42}$$

and since $\sigma_i(X) \geq 0$ we have $\sigma_i(Z) = \sqrt{\mu}$.

(ii) $\mu \leq \sigma_i^2(Z)$, which gives the optimization problem

$$\max_{\sigma_i(Z)} \mu - (\sigma_i(Z) - \sigma_i(X))^2, \tag{5.43}$$

which we can trivially solve by setting $\sigma_i(Z) = \sigma_i(X)$.

Together we get the optimal solution

$$\sigma_i(Z) = \max(\sqrt{\mu}, \sigma_i(X)) \, \forall i. \tag{5.44}$$

We now want to use this to derive the bi-conjugate by using that

$$\min(\mu, \sigma_i^2(Z)) = \min(\mu, \max(\mu, \sigma_i^2(X))) = \mu \tag{5.45}$$

and that

$$\|Z - X\|_F^2 = \sum_{i=1}^{n} (\sigma_i(Z) - \sigma_i(X))^2 = \\ \sum_{i=1}^{n} (\max(\sqrt{\mu}, \sigma_i(X)) - \sigma_i(X))^2 = \sum_{i=1}^{n} [\sqrt{\mu} - \sigma_i(X)]_+^2. \tag{5.46}$$

where

$$[x]_+ = \max(0, x). \tag{5.47}$$

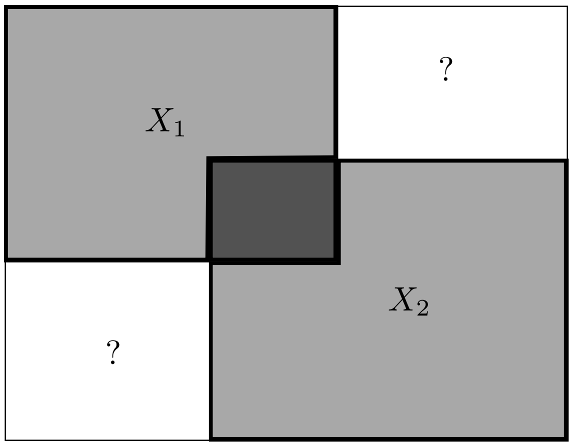


Figure 5.2: To handle missing data we divide the measurement matrix into sub-blocks. To recover the blocks with no data we must have an overlap between the blocks.

We can now finally derive the convex envelope of the original optimization problem (5.20). The convex envelope, or bi-conjugate is

$$f^{**}(X) = \|X - M\|_F^2 + \sum_{i=1}^{n} (\mu - [\sqrt{\mu} - \sigma_i(X)]_+^2). \tag{5.48}$$

Note that we also assume that the singular values are sorted in a decreasing order. To simplify notation later, we define

$$\mathcal{R}_\mu(X) = \sum_{i=1}^{n} (\mu - [\sqrt{\mu} - \sigma_i(X)]_+^2). \tag{5.49}$$

5.2.1 Missing Data

Even though we have now found a convex envelope of (5.20), it assumes that the measurement matrix is full, i.e. we have no missing data. For many computer vision problems this assumption is unrealistic and we need to tackle this problem. The key idea behind our approach is to divide the measurement matrix M into sub-blocks, where each sub-block is full. See Figure 5.2 for an illustration. Then we solve the optimization problem for each sub-block and when all approximations are found, we use these blocks to build the entire matrix X. Later we will see that the rank of the entire matrix is connected to the rank of the blocks we use to create X from.

Let R_i and C_i, $i = 1, \dots, K$ be a subset of row and column indices for each block i. We define the linear operator $\mathcal{P}_i : \mathbb{R}^{m \times n} \to \mathbb{R}^{|R_i| \times |C_i|}$, which extracts the elements at indices $R_i \times C_i$ and creates a sub-matrix of size $|R_i| \times |C_i|$ with no missing data.

Instead of trying to solve the original problem (5.18), we aim to solve

$$\min_X \sum_{i=1}^{K} \mu_i \operatorname{rank}(\mathcal{P}_i(X)) + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2, \tag{5.50}$$

by replacing it with its convex envelope

$$\min_X \sum_{i=1}^{K} \mathcal{R}_{\mu_i}(\mathcal{P}_i(X)) + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2. \tag{5.51}$$

By solving the optimization problem we end up with a bunch of sub-blocks. However, we have still not found the entire matrix X, but only a part of it. To recover the whole matrix we use the following lemma:

Lemma 5.2.1. Let X_1 and X_2 be two given matrices with overlap matrix X_{22} as shown in Figure 5.3 and let $r_1 = \operatorname{rank}(X_1)$ and $r_2 = \operatorname{rank}(X_2)$. Suppose that $\operatorname{rank}(X_{22}) = \min(r_1, r_2)$, then there exists a matrix X with $\operatorname{rank}(X) = \max(r_1, r_2)$. Additionally if $\operatorname{rank}(X_{22}) = r_1 = r_2$ then X is unique.

Proof. We will assume (w.l.o.g.) that $r_2 \leq r_1$ and look at the block X_2. The overlap X_{22} is of rank r_2 so there are r_2 linearly independent columns in $[X_{22}^T \, X_{32}^T]^T$ and rows in $[X_{22} \, X_{23}]$. Now the rank of X_2 is r_2 and we can find coefficient matrices C_1 and C_2 such that

$$\begin{bmatrix} X_{23} \\ X_{33} \end{bmatrix} = \begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix} C_1 \text{ and } \begin{bmatrix} X_{32} & X_{33} \end{bmatrix} = C_2 \begin{bmatrix} X_{22} & X_{23} \end{bmatrix}. \tag{5.52}$$

We therefore set $X_{13} := X_{12}C_1$ and $X_{31} := C_2X_{21}$. To determine the rank of the resulting X we first look at the number of linearly independent columns. By

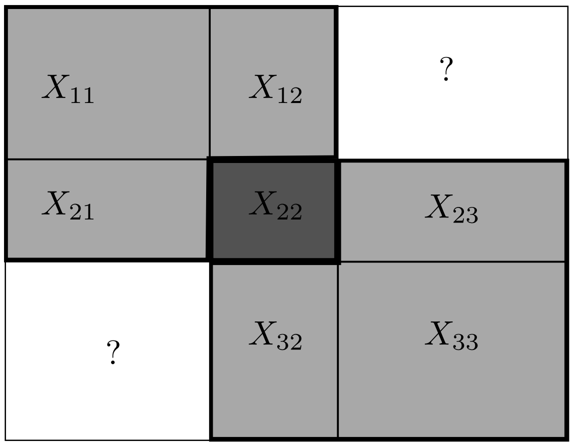


Figure 5.3: Two sub-blocks of a matrix with overlap X_{22}.

construction, the columns $[X_{13}^T\, X_{23}^T\, X_{33}^T]^T$ are linear combinations of the other columns and similarly, the rows $[X_{31}\, X_{32}\, X_{33}]$ are linear combinations of the other rows. Hence, the number of linearly independent columns (or rows) have not increased. Therefore X has the same rank as X_1.

If rank(X_{22}) = rank(X_1) = rank(X_2) = r, then C is unique. To prove this, assume that it is not unique, then

$$X_{13} = X_{12}C_1 \text{ and } \hat{X}_{13} = X_{12}\hat{C}_1. \tag{5.53}$$

Since both C_1 and $\hat{C}_1$ solves

$$\begin{bmatrix} X_{23} \\ X_{33} \end{bmatrix} = \begin{bmatrix} X_{22} \\ X_{32} \end{bmatrix} C \tag{5.54}$$

it follows that $C_1 - \hat{C}_1$ lies in the nullspace of

$$\begin{bmatrix} X_{22} \\ X_{32}. \end{bmatrix} \tag{5.55}$$

By assumption rank($[X_{12}^T\, X_{22}^T\, X_{32}^T]^T$) = rank($[X_{22}^T\, X_{32}^T]^T$). This implies that they share the same nullspace, so $C_1 - \hat{C}_1$ must lie in the nullspace of X_{12}. This gives $X_{13} = X_{12}C_1 = X_{12}\hat{C}_1 = \hat{X}_{13}$, a contradiction. □

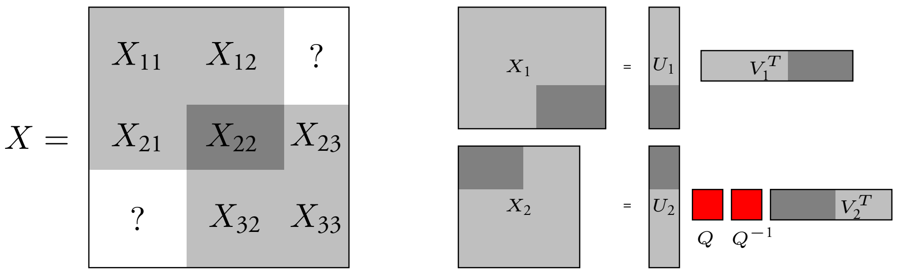


Figure 5.4: *Left:* The matrix X contains two overlapping blocks X_1 and X_2. The goal is to fill in the missing entries X_{13} and X_{31} such that rank(X) is kept to a minimum. *Right:* The low-rank factorizations of the two blocks X_1 and X_2. The overlap is marked in both the blocks and the factorizations.

The next step is now to extend the solution on the blocks to the entire matrix.We start by finding the maximal rank r_{max} among the blocks. Then we factorize X_1 and X_2 using SVD,

$$X_1 = U_1V_1^T \text{ and } X_2 = U_2V_2^T \text{ where } U_k \in \mathbb{R}^{m_k \times r_{max}}, V_k \in \mathbb{R}^{n_k \times r_{max}}. \tag{5.56}$$

However, we have an ambiguity here, because for any non-singular matrix Q, we have

$$U_1V_1^T = (U_1Q)(Q^{-1}V_1^T). \tag{5.57}$$

To resolve this, we consider the singular value decompositions of X_1 and X_2. Looking at Figure 5.4, we can see that we can create X_{22} from the sub-matrices by

$$X_{22} = \hat{U}_1\hat{V}_1^T \tag{5.58}$$

$$X_{22} = \hat{U}_2\hat{V}_2^T, \tag{5.59}$$

where $\hat{U}_i\hat{V}_i^T$ is the restriction of the $U_iV_i^T$ to X_{22}.

Due to ambiguity, $\hat{U}_1$ will in general not equal $\hat{U}_2$ and $\hat{V}_1$ will be different from $\hat{V}_2$. Instead we try to find an invertible transformation Q which transforms $\hat{U}_1$ into $\hat{U}_2$, i.e. we seek to solve

$$\hat{U}_1 = \hat{U}_2Q \tag{5.60}$$

$$\hat{V}_1 = Q^{-1}\hat{V}_2. \tag{5.61}$$

In this case we solve the optimization problem

$$\min_{Q} \quad \|\hat{U}_1 + \hat{U}_2 Q\|_F^2 + \|Q\hat{V}_1 - \hat{V}_2\|_F^2 \tag{5.62}$$

in a least square sense. With this Q, we can concatenate U_1 and U_2 to $U = [U_1 Q^{-1}, \tilde{U}_2]^T$ where $\tilde{U}_2$ contains the elements which are not common to $U_1 Q^{-1}$. The same way we construct V and obtain a complete matrix X by

$$X = UV^T. \tag{5.63}$$

This way we can iteratively combine the sub-blocks and create a single matrix X and thanks to Lemma 5.2.1, we know that our matrix X does not have higher rank then the rank among the sub-matrices.

5.3 Optimization

To actually find a low rank approximation of each sub-block, we must do some optimization. If we would optimize each sub-block independently, we would have no guarantee that they would agree on the overlap. Thus, we must enforce consistency on the overlap as a constraint. This results in the (convex) optimization problem

$$\sum_{i=1}^{K} \mathcal{R}_{\mu_i}(X_i) + \|X_i - \mathcal{P}_i(M)\|_F^2 \tag{5.64}$$

$$\text{subject to}$$

$$\mathcal{P}_i(X) = X_i \quad \forall i = 1...K.$$

An ADMM [11] formulation results in the augmented Lagrangian

$$\sum_{i=1}^{K} \mathcal{R}_{\mu_i}(X_i) + \|X_i - \mathcal{P}_i(M)\|_F^2 + \rho\|X_i - \mathcal{P}_i(X) + \Lambda_i\|_F^2 - \rho\|\Lambda_i\|_F^2. \tag{5.65}$$

At each iteration we solve and update the variables by

$$X_i^{t+1} = \arg\min_{X_i} \quad \mathcal{R}_{\mu_i}(X_i) + \|X_i - \mathcal{P}_i(M)\|_F^2 + \rho\|X_i - \mathcal{P}_i(X^t) + \Lambda_i^t\|_F^2, \tag{5.66}$$

for $i = 1...K$ and

$$X^{t+1} = \arg\min_X \sum_{i=1}^{K} \rho \|X_i^{t+1} - \mathcal{P}_i(X) + \Lambda_i^t\|_F^2 \tag{5.67}$$

$$\Lambda_i^{t+1} = \Lambda_i^t + X_i^{t+1} - \mathcal{P}_i(X^{t+1}). \tag{5.68}$$

5.3.1 Proximal Operator

Finding X^{t+1} is a simple least squares problem, whereas finding the optimal sub-block X_i^{t+1} is more complicated. We need to solve

$$\min_{X_i} F(X_i) \tag{5.69}$$

where

$$F(X_i) = \mathcal{R}_{\mu_i}(X_i) + \|X_i - \mathcal{P}_i(M)\|_F^2 + \rho\|X_i - \mathcal{P}_i(X) + \Lambda_i\|_F^2. \tag{5.70}$$

This objective can be rewritten as

$$\begin{aligned}&\sum_{j=1}^{N}(\mu - [\sqrt{\mu} - \sigma_j(X_i)]_+^2) + (1+\rho)\|X\|_F^2 \\ &\qquad - 2\langle \mathcal{P}_i(M) + \rho(\mathcal{P}_i(X) - \Lambda_i), X_i\rangle + \rho\|\mathcal{P}(X) - \Lambda_i\|_F^2\end{aligned} \tag{5.71}$$

and simplified to

$$F(X_i) = G(X_i) - 2\langle Y, X_i\rangle. \tag{5.72}$$

where

$$G(X_i) = \sum_{j=1}^{n}(-[\sqrt{\mu} - \sigma_j(X_i)]_+^2) + (1+\rho)\|X_i\|_F^2 \tag{5.73}$$

and

$$Y = \mathcal{P}_i(M) + \rho(\mathcal{P}_i(X^t) - \Lambda_i^t). \tag{5.74}$$

Since F is convex, it is sufficient to find where $0 \in \partial F$.

For this we need some theory about unitary invariant matrix functions, but we start by defining the function

$$g_i(\sigma) = -[\sqrt{\mu} - |\sigma|]_+^2 + (1+\rho)\sigma^2. \tag{5.75}$$

To see that $g_i(\sigma)$ is convex, one sees that g_i is a special case of $f^{**}(X)$, where $X \in \mathbb{R}$. Since the bi-conjugate $f^{**}(X)$ is convex, g_i must be convex. With our definition of g_i, we can now define

$$g(\boldsymbol{\sigma}) = \sum_{i=1}^{n} g_i(\sigma_i), \tag{5.76}$$

which is absolutely symmetric and convex, since g_i is convex for all $i = 1...n$. Now $G(X) = g \circ \boldsymbol{\sigma}(X)$, where

$$\boldsymbol{\sigma}(X) = (\sigma_1(X), \sigma_2(X), ..., \sigma_n(X))^T. \tag{5.77}$$

To derive the subgradients we make use of the following lemma in [46]:

Lemma 5.3.1. (Characterization of Subgradients) Let us suppose that the function $f : \mathbb{R}^q \to (-\infty, \infty]$ is absolutely symmetric and that the $m \times n$ matrix X has $\sigma(X)$ in dom f. Then the $m \times n$ matrix Y lies in $\partial(f \circ \sigma)(X)$ if and only if $\sigma(Y)$ lies in $\partial f \circ \sigma(X)$ and there exists a simultaneous singular value decomposition of the form

$$X = V(\text{Diag}(\sigma(X)))U, \tag{5.78}$$

$$Y = V(\text{Diag}(\sigma(Y)))U \tag{5.79}$$

where U and V are unitary matrices. In fact

$$\partial(f \circ \sigma)(X) = \{V(\text{Diag}(\mu))U | \mu \in \partial f(\sigma(X)), X = V(\text{Diag}(\sigma(X)))U\}. \tag{5.80}$$

To compute the subdifferential of $G(X)$, we compute the subdifferential of $g(\boldsymbol{\sigma}(X))$ instead. Since $g(\boldsymbol{\sigma}(X))$ is a sum of one-dimensional functions we can compute the subgradient by treating each term individually. For each g_i we have

$$\partial g_i(\sigma) = \begin{cases} 2\text{sgn}(\sigma)[\sqrt{\mu} - |\sigma|]_+ + 2(1+\rho)\sigma & \sigma \neq 0 \\ [-2\sqrt{\mu}, 2\sqrt{\sigma}] & \sigma = 0. \end{cases} \tag{5.81}$$

Now, the aim is to solve $0 \in \partial F(X)$, which is equivalent of solving $2Y \in \partial G(X)$. Recall that $F(X) = G(X) - 2\langle X, Y \rangle$.

If $Y = U\text{Diag}(\sigma(Y))V^T$ then we can verify that $X = U\text{Diag}(\sigma(X))V^T$ where

$$\sigma_i(X) = \begin{cases} \frac{\sigma(Y)}{1+\rho} & \text{if } \sigma_i(Y) \geq (1+\rho)\sqrt{\mu} \\ \frac{\sigma_i(Y)-\sqrt{\mu}}{\rho} & \text{if } \sqrt{\mu} \leq \sigma_i(Y) \leq (1+\rho)\sqrt{\mu} \\ 0 & \text{if } \sigma_i(Y) \leq \sqrt{\mu} \end{cases}, \tag{5.82}$$

is such that $2Y \in \partial G(X)$. Since

$$\partial G(X) = U\text{diag}(\partial g \circ \boldsymbol{\sigma}(X))V^T \tag{5.83}$$

we have to check that

$$2\sigma_i(Y) \in \partial g_i(\sigma_i(X)) \tag{5.84}$$

for all i. Now there are three cases to check:

Case 1:

$$\sigma_i(Y) \geq (1+\rho)\sqrt{\mu} : \tag{5.85}$$

$$\sigma_i(X) = \frac{\sigma_i(Y)}{1+\rho} \tag{5.86}$$

gives

$$\frac{\partial g_i}{\partial \sigma}\left(\frac{\sigma_i(Y)}{1+\rho}\right) = 2\left[\sqrt{\mu} - \frac{\sigma_i(Y)}{1+\rho}\right]_+ + 2(1+\rho)\frac{\sigma_i(Y)}{1+\rho} = 2\sigma_i(Y). \tag{5.87}$$

Case 2:

$$\sqrt{\mu} \leq \sigma_i(Y) \leq (1+\rho)\mu : \tag{5.88}$$

$$\sigma_i(X) = \frac{(\sigma_i(Y) - \sqrt{\mu})}{\rho} \tag{5.89}$$

gives

$$\frac{\partial g_i}{\partial \sigma}\left(\frac{(\sigma_i(Y) - \sqrt{\mu})}{\rho}\right) = \tag{5.90}$$

$$2\left(\sqrt{\mu} - \frac{(\sigma_i(Y) - \sqrt{\mu})}{\rho}\right) + 2(1+\rho)\frac{(\sigma_i(Y) - \sqrt{\mu})}{\rho} = 2\sigma_i(Y) \tag{5.91}$$

If $\frac{(\sigma_i(Y)-\sqrt{\mu})}{\rho} = 0$, then

$$\frac{\partial g_i}{\partial \sigma}(0) = 2\sqrt{\mu} = 2\sigma_i(Y), \tag{5.92}$$

and it is in (5.81).

Case 3:

$$\sigma_i(Y) \leq \sqrt{\mu} \tag{5.93}$$

Here $\sigma_i(X) = 0$ and therefore $2\sigma_i(Y) \leq 2\sqrt{\mu}$ is contained in the subdifferential (5.81).

5.4 Experiments

To evaluate this method we start by using synthetic data and compare the results with other methods, such as the nuclear norm and other local optimization methods. First the convex relaxation is evaluated using synthetic data. We define

$$f(X) = \sum_{i=1}^{K} \mu_i \operatorname{rank}(\mathcal{P}_i(X)) + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2 \tag{5.94}$$

and our derived convex relaxation

$$f_{\mathcal{R}}(X) = \sum_{i=1}^{K} \mathcal{R}_{\mu_i}(\mathcal{P}_i(X)) + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2. \tag{5.95}$$

Then random rank 3 matrices were generated of dimension 100×100 by sampling $U, V \in \mathbb{R}^{100\times 3}$ from a Gaussian distribution with zero mean and unit variance. Then M was formed by $M = UV^T$. The observation matrix W was chosen to be a band-diagonal matrix with bandwidth 40 similar to Figure 5.5. The blocks were laid out such that the overlap was 6×6 and contained no missing data. Then the solution

$$X_{\mathcal{R}}^* = \arg\min_X f_{\mathcal{R}}(X), \tag{5.96}$$

where μ_i was set to 1 for every i was found. To M, we added different levels of Gaussian noise and the test was repeated 1000 times for each noise level. In

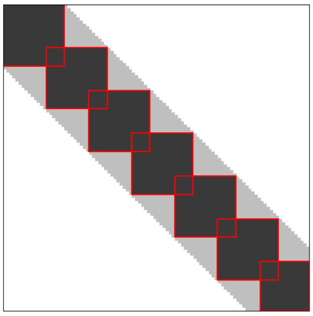


Figure 5.5: The pattern in the synthetic experiments and the overlap.

Figure 5.6 we plot the average error for each noise level of $f(X^*_{\mathcal{R}})$ and $f_{\mathcal{R}}(X^*_{\mathcal{R}})$, note that if $f(X^*_{\mathcal{R}}) = f_{\mathcal{R}}(X^*_{\mathcal{R}})$, then the global minimizer is found.

To compare with other methods we substitute the rank function with the nuclear norm. The nuclear norm is also convex, so it can also be used in our block decomposition framework. Hence, the same experiment was made with the objective

$$f_N(X) = \sum_{i=1}^{K} \mu_i \|\mathcal{P}_i(X)\|_* + \|\mathcal{P}_i(X) - \mathcal{P}_i(M)\|_F^2. \tag{5.97}$$

The results are shown in the top graph of Figure 5.6. Note that the constraint $\sigma_{max}(\mathcal{P}_i(X)) \leq 1$ can be violated, so f_N is not necessarily a lower bound on f.

However, there also exist other methods for obtaining low-rank approximations. We compare to two non-convex methods, namely OptSpace [42] and Truncated Nuclear Norm Regularization (TNNR), [35]. Both OptSpace and TNNR are local methods, that is, they are not convex. OptSpace is based on local optimization on Grassmanian manifolds and in TNNR an energy which penalizes the last $(n - r)$ singular values is minimized. The experiments were made the same way as above with randomly chosen matrices $U, V \in \mathbb{R}^{3\times 100}$. Both these approaches try to estimate a fixed rank approximation, that is the rank is set before hand. In contrast our method is a trade-off between the rank and the data-term $\|X - M\|_F^2$. We therefore iterate our method over μ_i to get the same rank. The average values of $\|W \odot (X - M)\|_F^2$ are shown in 5.6. Even though the plots suggests that our method is better, often both OptSpace and TNNR gives similar or better results than our method. However, local minima result in a higher average than our method, but on the other hand, since OptSpace and TNNR minimizes the original function directly, their solution will be at least as good as ours when it does not get stuck in a local minimum.

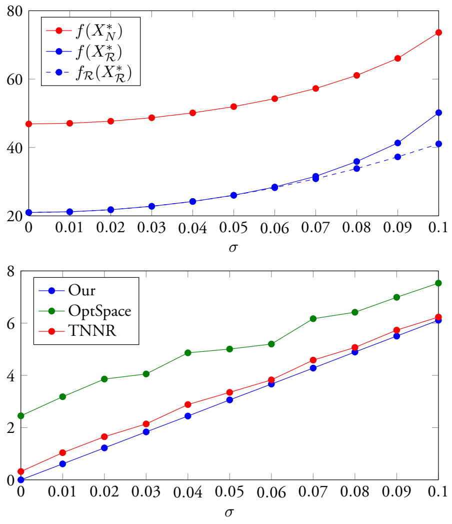
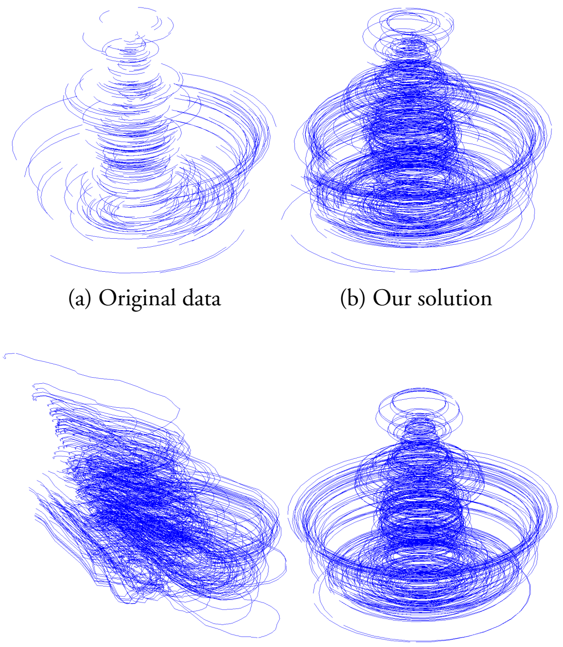


Figure 5.6: Evaluation of the proposed formulation on synthetic data with varying noise levels. *Top*: Comparison of our convex relaxation solution $X_\mathcal{R}^*$ with the nuclear norm X_N^*. *Bottom*: The error $\|W \odot (X - M)\|_F$ for varying noise levels.

5.4.1 Real Data

As described earlier in this chapter, one application for low-rank approximation is rigid structure from motion. We test our method on the well-known Oxford dinosaur sequence. The measurement matrix M will contain the tracked 2D coordinates of the tracked 3D points, as described in the beginning of this chapter. Since we cannot handle outliers, we choose an outlier free subset of this dataset consisting of 321 3D points where each point is seen in at least six images. The observation matrix W is shown in the left of Figure 5.8 and clearly demonstrates the band diagonal pattern for these structure from motion problems. For comparison we also solve the problem using the nuclear norm and bundle adjustment.



(a) Original data (b) Our solution

(c) Nuclear norm (d) Perspective projection

Figure 5.7: *Top Left*: Observed image point trajectories. *Top Right*: Result obtained with our method. *Bottom Left*: Result obtained with the nuclear norm. *Bottom Right*: Perspective reconstruction followed by projecting onto rank 4 using SVD.

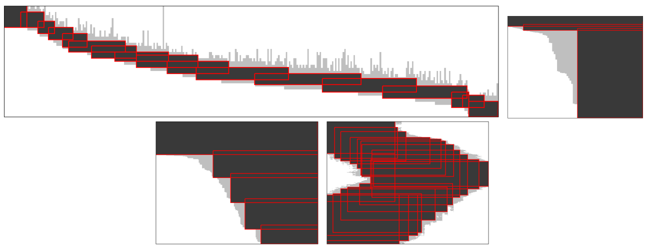


Figure 5.8: Structured data patterns of observations (matrix W) and sub-blocks for the *dino*, *book*, *hand* and *banner* sequences.

The nuclear norm formulation is

$$\min_{X} \mu\|X\|_* + \|W \odot (X - M)\|_F^2 \tag{5.98}$$

which we optimize with the shrinkage operator and ADMM. The results are show in Figure 5.7.

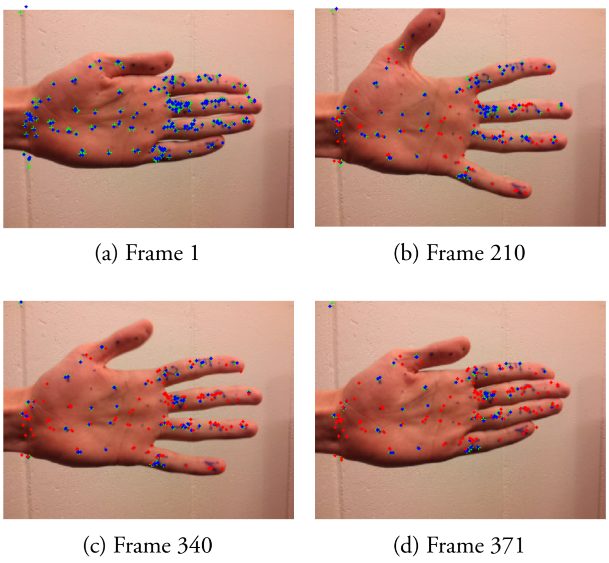
As seen in the figure, the nuclear norm performs much worse than our method. This clearly shows the effect of penalizing all singular values rather than just the singular values smaller than $\sqrt{\mu}$. The perspective projection gives a more visually appealing result. The errors $||W \odot (X - M)||_F$ for the three solutions were 73.2 (our), 1902.5 (nuclear) and 116.2 (perspective), so even if the perspective projection looks better, our solution is a better low-rank approximation of the original data.

5.4.2 Linear Shape Models

Another application is linear shape models, which are common in non-rigid structure from motion. Let M^f be the 2D- or 3D-coordinates of N tracked points in frame f. The model we have assumes that in each frame the coordinates are a linear combination of some unknown shape basis vectors, i.e.

$$M^f = \sum_{k=1}^{K} c_k^f \mathbf{s}_k, \quad f = 1 \dots F, \tag{5.99}$$

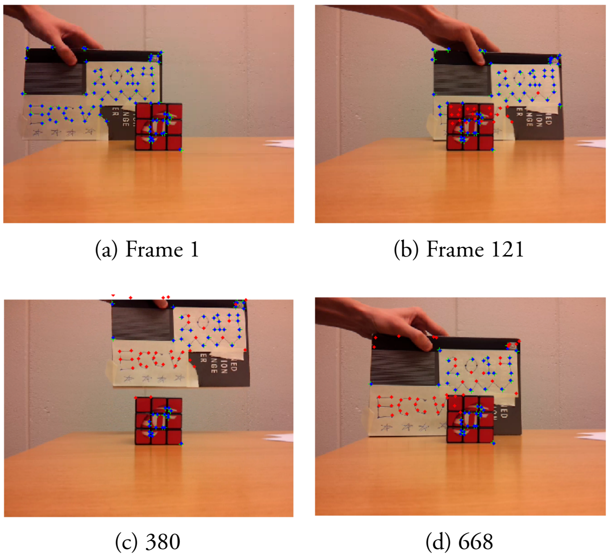
where $\mathbf{s}_k \in \mathbb{R}^{2\times N}$ is the shape basis and c_k^f is the coefficient. We do not want more basis elements than necessary to describe the movement. Consequently, we



(a) Frame 1 (b) Frame 210

(c) Frame 340 (d) Frame 371

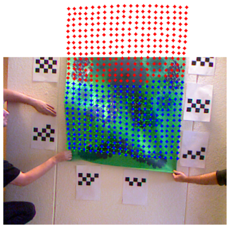
Figure 5.9: Frames 1, 210, 340 and 371 of the *hand* sequence. The solution has rank 5.

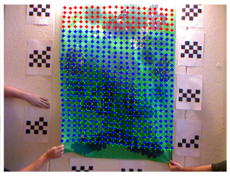


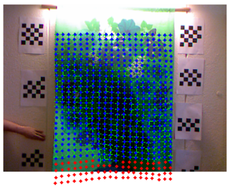
(a) Frame 1 (b) Frame 121

(c) 380 (d) 668

Figure 5.10: Frames 1, 121, 380 and 668 of the *book* sequence. The solution has rank 3.


(a) Frame 70


(b) Frame 155


(c) Frame 357

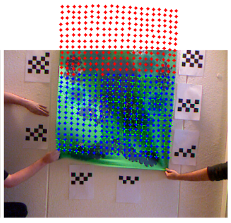
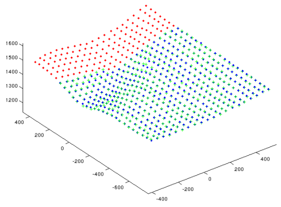
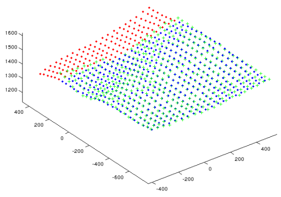
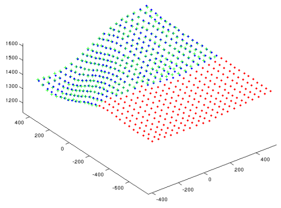

(d) Frame 650

Figure 5.11: Frames 70, 155, 357 and 650 of the *banner* sequence. The solution has rank 9.


(a) Frame 329 our solution.


(b) Frame 329 nuclear norm.


(c) Frame 650 our solution.

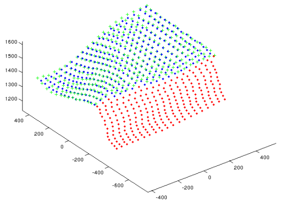

(d) Frame 650 nuclear norm.

Figure 5.12: *From top left to bottom right:* Our solution (frame 329), nuclear norm solution (frame 329), our solution (frame 650), nuclear norm solution (frame 650).

seek a low-rank approximation of the measurement matrix M. The measurement matrix M is obtained by stacking all tracked points for each frame on top of each other. In other words, the first two rows in M are the tracked points for frame 1 and so on. The elements of the observation matrix W indicates if the point was successfully tracked in that frame or not. In the first two datasets, *Book* and *Hand*, we track points through a video sequence using the standard KLT-tracker [47]. Due to tracking failure and occlusion, we get missing data which forms the patterns shown in Figure 5.8 where the selected blocks are shown as well. Using (5.51), we find a low rank approximation for each dataset. The results can be seen in Figure 5.9 and Figure 5.10. The blue points are the reconstructed points that were successfully tracked through the entire sequence. The red dots are the reconstructed missing data and the green crosses are the original measured data. As can be seen in both sequences, the derived solution gives reasonable results.

In the third experiment, we used a Kinect camera to record a video of a moving piece of fabric. This gives a 3D grid of points and missing data comes from a limited field of view and missing depth values in the depth image. To register all points in a common coordinate system, we track the cameras using the patterns on the wall (Figure 5.11). The obtained solution can be seen in Figure 5.11 and Figure 5.12. For comparison, the results for the nuclear norm is also provided. It is clear that our solution yields a more realistic reconstruction of the movement, in contrast to the nuclear norm which has a bias toward small singular values.

5.5 Conclusion

In this chapter we derived a convex envelope of the rank-function plus a data-term. This could then be combined with other convex constraints and optimized in a ADMM-framework. Results suggest that our method works better in comparison to the nuclear norm which is another popular convex relaxation.

Chapter 6

Minimizing the Maximal Rank

6.1 Introduction

In this chapter we will see that we can formulate an energy function where we optimize over several matrices simultaneously and favor solutions where all matrices have the same rank. In Section 5.2 we saw that the convex envelope could be derived for the function

$$f(X) = \mu \operatorname{rank}(X) + \|X - M\|_F^2. \tag{6.1}$$

By dividing the measurement matrix M into sub-blocks with no missing data it was possible to derive a convex optimization problem with the linear constraint that the overlap of the sub-matrices should coincide.

In this part we consider the case where we seek to find a low rank estimation of several matrices simultaneously, where all matrices should have the same rank.

This is a natural formulation if one wants to approximate a manifold by tangent spaces. Consider a d-dimensional connected manifold in $\mathbb{R}^N$. Locally, this manifold can be approximated with a d-dimensional tangent space. Therefore, approximating the data with a manifold can be thought of as locally approximating data with a low-rank matrix, all of rank d. An example of the idea is illustrated in Figure 6.1. A practical application of this can be denoising where several images are used simultaneously. For example [33] consider each vectorized image as a point in $\mathbb{R}^N$. The idea is that all images that are similar to each other will be close in $\mathbb{R}^N$ and lie in the same tangent space. Due to noise that assumption will not be fulfilled, but hopefully one can find the d-dimensional tangent space which contains the pure images and extract them to get the denoised images. By using several images, the goal is to extract information from all those images simultaneously to get a better denoising.

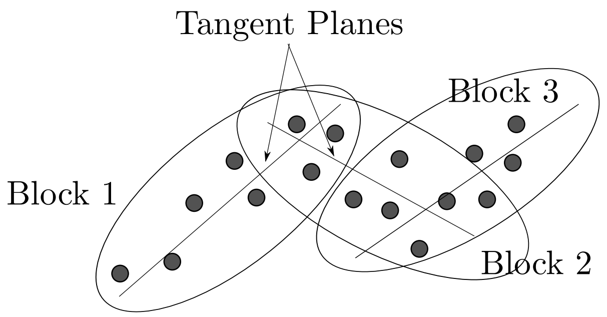


Figure 6.1: A set of data points that lies on a d-dimensional manifold can be locally approximated by tangent spaces.

This idea could also be extended to single image denoising. Instead of having several similar images that should span a tangent space, one can divide the image into several overlapping patches. Then patches similar to each other should lie close in space and span a lower-dimensional tangent space where there is less noise.

By assumption, all tangent spaces should have the same dimension d. This leads naturally to a set of matrices $\hat{X} = \{X_1, \ldots, X_b\}$, where all have rank d.

The optimization problem that will be derived will only have one parameter to control the rank of the matrices. This can be a practical advantage also in applications which are not directly related to approximation of manifolds. For example in structure from motion or linear shape basis estimation where we might know the rank of the matrices we want to find, each sub-block has a parameter μ_i that has to be tuned in Section 5.2. That might be a practical obstacle and then it might be favorable to use the procedure which will be derived in this chapter.

6.2 Theory

We will now derive the new objective function and its convex envelope. Let $\hat{X} = (X_1, \ldots, X_b)$ be a collection of matrices. The scalar product is denoted as

$$\langle \hat{X}, \hat{Y} \rangle = \sum_{i=1}^{b} \langle X_i, Y_i \rangle = \sum_{i=1}^{b} \operatorname{tr}(X_i^T Y_i) \tag{6.2}$$

and the function $r(\hat{X})$ is defined as

$$r(\hat{X}) = \max(\text{rank}(X_1), \dots, \text{rank}(X_b)). \tag{6.3}$$

The objective function will be on the form

$$\min \mu\, r(\hat{X}) + \|\hat{X} - \hat{M}\|^2 \tag{6.4}$$

where

$$\|\hat{X} - \hat{M}\|^2 = \sum_{i=1}^{b} \|X_i - M_i\|_F^2. \tag{6.5}$$

The objective (6.4) is a formulation that favors solutions $\hat{X}$ such that all submatrices have the same rank. To see this, assume that $\hat{X}'$ is a possible solution to (6.4), where we have $\text{rank}(X_i') < r(\hat{X}')$ for some X_i' among $(X_1', \dots, X_b')$ in the possible solution $\hat{X}'$. Then the rank of X_i' can be increased to equal $r(\hat{X}')$ without increasing the term $r(\hat{X}')$ in (6.4). On the other hand, by increasing the rank of X_i' the data term $\|\hat{X}' - \hat{M}\|^2$ will decrease. Consequently, we can get a better fit to the data without increasing the rank-penalty.

6.2.1 Convex Envelope

Just like in Section 5.2, the convex envelope of our objective is derived by working with Fenchel Conjugates. In this case the Fenchel Conjugate is

$$f^*(\hat{Y}) = \max_{\hat{X}} \langle \hat{X}, \hat{Y} \rangle - \mu\, r(\hat{X}) - \|\hat{X} - \hat{M}\|^2. \tag{6.6}$$

By completing squares we obtain

$$f^*(\hat{Y}) = \max_{\hat{X}} -\mu r(\hat{X}) - \|\hat{X} - \hat{M} - \frac{\hat{Y}}{2}\|^2 + \|\hat{M} + \frac{\hat{Y}}{2}\|^2 - \|\hat{M}\|^2. \tag{6.7}$$

By setting $\hat{Z} = \hat{M} + \frac{\hat{Y}}{2}$ the maximization in (6.4) can be written as

$$\max_{\hat{X}} -\mu r(\hat{X}) - \|\hat{X} - \hat{Z}\|^2 + \|\hat{Z}\|^2 - \|\hat{M}\|^2. \tag{6.8}$$

and by rewriting the maximization we obtain

$$\max_{k} \max_{\substack{\hat{X} \\ r(\hat{X})=k}} -\mu k - \|\hat{X} - \hat{Z}\|^2 + \|\hat{Z}\|^2 - \|\hat{M}\|^2. \tag{6.9}$$

For a fixed maximal rank k, the optimal solution for each matrix X_j is given by

$$X_j = \sum_{i=1}^{k} \sigma_i(Z_j)\mathbf{u}_i\mathbf{v}_i^T. \tag{6.10}$$

Here $\sigma_i(Z_j)$ is the i-th singular value of Z_j and $\mathbf{u}_i$ and $\mathbf{v}_i$ are the singular vectors of Z_j and we assume $\sigma_i(Z_j) \geq \sigma_{i+1}(Z_j)$. Plugging this into (6.9) gives

$$\max_{k} -\mu k - \sum_{i=k+1}^{n} \|\sigma_i(\hat{Z})\|_2^2 + \|\hat{Z}\|^2 - \|\hat{M}\|^2. \tag{6.11}$$

Where $\sigma_i(\hat{Z})$ is the vector $(\sigma_i(Z_1), \ldots, \sigma_i(Z_b))$ and $\|\sigma_i(\hat{Z})\|_2$ is the standard euclidean vector norm. By observing that

$$-\mu k - \sum_{i=k+1}^{n} \|\sigma_i(\hat{Z})\|_2^2 = -\sum_{i=1}^{k} \mu - \sum_{i=k+1}^{n} \|\sigma_i(\hat{Z})\|_2^2 \tag{6.12}$$

we conclude that the maximizing k should be chosen such that

$$\|\sigma_{k+1}(\hat{Z})\|_2^2 \leq \mu \leq \|\sigma_k(\hat{Z})\|_2^2. \tag{6.13}$$

The conjugate function is thus

$$f^*(\hat{Y}) = -\sum_{i=1}^{n} \min(\mu, \|\sigma_i(\hat{Z})\|_2^2) + \|\hat{Z}\|^2 - \|\hat{M}\|^2. \tag{6.14}$$

Recall that $\hat{Z}$ depends on $\hat{Y}$ through $\hat{Z} = \hat{M} + \frac{\hat{Y}}{2}$. The next step is now to consider the biconjugate

$$f^{**}(\hat{X}) = \max_{\hat{Y}} \langle \hat{X}, \hat{Y} \rangle - f^*(\hat{Y}) \tag{6.15}$$

$$= \max_{\hat{Z}} 2\langle \hat{X}, \hat{Z} - \hat{M} \rangle - f^*(2\hat{Z} - 2\hat{M}). \tag{6.16}$$

Using our derived expression in (6.14) and plugging it into (6.16) the biconjugate can be written as

$$f^{**}(\hat{X}) = \max_{\hat{Z}} \; 2\langle \hat{X}, \hat{Z} - \hat{M} \rangle - (-\sum_{i=1}^{n} \min(\mu, \|\sigma_i(\hat{Z})\|_2^2) + \|\hat{Z}\|^2 - \|\hat{M}\|^2) \tag{6.17}$$

$$= \max_{\hat{Z}} \; 2\langle \hat{X}, \hat{Z} - \hat{M} \rangle + \sum_{i=1}^{n} \min(\mu, \|\sigma_i(\hat{Z})\|_2^2) - \|\hat{Z}\|^2 + \|\hat{M}\|^2. \tag{6.18}$$

By completing squares (6.18) becomes

$$f^{**}(\hat{X}) = \max_{\hat{Z}} \sum_{i=1}^{n} \min(\mu, \|\sigma(\hat{Z})\|_2^2) - \|\hat{X} - \hat{Z}\|^2 + \|\hat{X} - \hat{M}\|^2. \tag{6.19}$$

Using von Neumann's trace theorem the maximizing Z_j must have an SVD with the same U and V as X_j. Therefore, the convex envelope is a maximization over the singular values of Z_j, $j = 1, \ldots, b$:

$$f^{**}(\hat{X}) = \max_{\hat{Z}} \sum_{i=1}^{n} \min(\mu, \|\sigma_i(\hat{Z})\|_2^2) - \|\sigma_i(\hat{Z}) - \sigma_i(\hat{X})\|_2^2 + \|\hat{X} - \hat{M}\|^2. \tag{6.20}$$

The function $f^{**}(\hat{X})$ thus involves a maximization over $\hat{Z}$ for which we have not found a closed form solution. Luckily, the proximal operator can be computed using a single cone program. The proximal operator is defined as

$$\mathrm{prox}_{f^{**}}(\hat{Y}) = \arg\min_{\hat{X}} \; f^{**}(\hat{X}) + \rho\|\hat{X} - \hat{Y}\|^2 \tag{6.21}$$

which is the basis for ADMM. The trick is to switch the order of minimization and maximization and thereby obtain a closed form solution for $\hat{X}$. If $\rho > 0$ the objective function is closed, proper convex-convcave and continuous and optimization can be restricted to a compact set. Switching optimization order is therefore justified by the existence of a saddle point, see [60].

To minimize (6.21) with respect to $\hat{X}$ we only consider the terms containing $\hat{X}$

$$-\|\hat{X} - \hat{Z}\|^2 + \|\hat{X} - \hat{M}\|^2 + \rho\|\hat{X} - \hat{Y}\|^2. \tag{6.22}$$

Differentiating with respect to $\hat{X}$ gives

$$-(\hat{X}-\hat{Z})+(\hat{X}-\hat{M})+\rho(\hat{X}-\hat{Y})=0 \tag{6.23}$$

$$\hat{X}=\frac{\hat{M}-\hat{Z}}{\rho}+\hat{Y}. \tag{6.24}$$

Inserting this into (6.22) gives

$$-\|\hat{Z}-(\frac{\hat{M}-\hat{Z}}{\rho}+\hat{Y})\|^2+\|\frac{\hat{M}-\hat{Z}}{\rho}+\hat{Y}-\hat{M}\|+\rho\|\frac{\hat{M}-\hat{Z}}{\rho}+\hat{Y}-\hat{Y}\|^2 \tag{6.25}$$

which after completing squares turns out to be

$$-\frac{\rho+1}{\rho}\|\hat{Z}-\hat{W}\|^2+C, \tag{6.26}$$

where

$$\hat{W}=\frac{\rho\hat{Y}+\hat{M}}{\rho+1} \tag{6.27}$$

and C contains all terms not depending on $\hat{Z}$. Since we are only interested in the optimizers $\hat{Z}$ and $\hat{X}$, we can ignore C.

The objective function thus becomes

$$\max_{\hat{Z}}\sum_{i=1}^{n}\min(\mu,\|\sigma_i(\hat{Z})\|^2)-\frac{\rho+1}{\rho}\|\hat{Z}-\hat{W}\|^2. \tag{6.28}$$

The terms in the sum only depends on the singular values of Z_j. The second term can be written

$$\|\hat{Z}-\hat{W}\|^2=\|\hat{Z}\|^2-\sum_{j=1}^{b}\langle Z_j,W_j\rangle+\|\hat{W}\|^2. \tag{6.29}$$

We see that by von Neumann's trace theorem the Singular Value Decomposition, SVD, of Z_j shall have the same U and V as the SVD of W_j in order to minimize the second term. This leads to the maximization problem

$$\max_{\hat{Z}}\sum_{i=1}^{n}\min(\mu,\|\sigma_i(\hat{Z})\|^2)-\frac{\rho+1}{\rho}\|\sigma_i(\hat{Z})-\sigma_i(\hat{W})\|^2. \tag{6.30}$$

To find the optimal singular values of $\hat{Z}$ we will show that this can be solved using a cone program. By introducing the auxiliary variables s_i, $i = 1, \dots, n$ and writing

$$\max \sum_{i=1}^{n} s_i \tag{6.31}$$

$$\text{s.t. } s_i \leq \mu - \frac{\rho+1}{\rho} \|\sigma_i(\hat{Z}) - \sigma_i(\hat{W})\|_2^2 \tag{6.32}$$

$$s_i \leq \|\sigma_i(\hat{Z})\|_2^2 - \frac{\rho+1}{\rho} \|\sigma_i(\hat{Z}) - \sigma_i(\hat{W})\|_2^2 \tag{6.33}$$

we get a convex cone problem.

As we are maximizing the sum s_i, (6.32) or (6.33) will obtain equality. Another constraint we have is that the singular values shall be decreasing for each block. Hence, we add a linear constraint on the singular values which results in the formulation

$$\max \sum_{i=1}^{n} s_i \tag{6.34}$$

$$\text{s.t. } \|\sigma_i(\hat{Z}) - \sigma_i(\hat{W})\|_2^2 \leq \frac{\rho}{\rho+1}(\mu - s_i) \tag{6.35}$$

$$\|\sigma_i(\hat{Z}) - (\rho+1)\sigma_i(\hat{W})\|_2^2 \leq \rho(\|(\rho+1)\sigma_i(\hat{W})\|^2 - s_i) \tag{6.36}$$

$$\sigma_1(\hat{Z}) \succeq \sigma_2(\hat{Z}) \succeq \dots \sigma_n(\hat{Z}) \succeq 0. \tag{6.37}$$

The inequalities (6.35) and (6.36) can be realized using the convex cone

$$\{(x_1, x_2, x_3); x_1 x_2 \geq x_3^2,\ x_1 + x_2 \geq 0\}. \tag{6.38}$$

To solve (6.37) we use standard solvers like [69] and [2].

6.3 Experiments

6.3.1 Manifold Denoising

The first application is Manifold denoising. In Manifold denoising one assumes that all images represented as column vectors lie on a manifold. If the images are corrupted by noise, they will deviate from the manifold. By estimating the

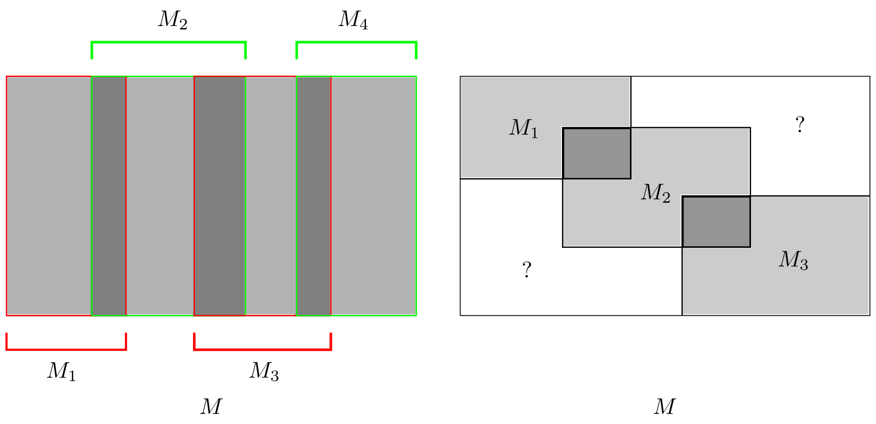


Figure 6.2: Example of how a measurement matrix can be divided into blocks. *Left:* Example of blocks for tangent spaces. *Right:* Block division with missing data.

underlying manifold from the noisy data, one should get a less noisy set of images. In this section, the goal is to estimate the underlying manifold by approximating it with tangent spaces at each image point.

To do this, we first attach a set of neighbors to each image point. The neighborhood for an image $\mathbf{m}$ are the K closest images in euclidean distance, including $\mathbf{m}$ itself. By using the associated neighborhood, we can approximate the tangent space at $\mathbf{m}$. By stacking all the neighbors of $\mathbf{m}$ in a measurement matrix M and computing the SVD of M we can write M as

$$M = \sum_{i=1}^{n} \mathbf{u}_i \mathbf{v}_i^T \sigma_i(M). \tag{6.39}$$

The idea is that singular vectors that corresponds to high singular values contain more signal and that singular vectors corresponding to small singular values contain more noise.

Since we assume that M is noisy, we use less than n singular vectors to approximate M:

$$X = \sum_{i=1}^{k<n} \mathbf{u}_i \mathbf{v}_i^T \sigma_i(M). \tag{6.40}$$

X should then be less noisy compared to M. The k singular vectors $\mathbf{u_i}$ spans the approximated tangent space of the manifold around $\mathbf{m}$.

Generalizing this we see that each image $\mathbf{m}_i$ gives a neighborhood M_i so we get a collection of measurement matrices $\hat{M}$. By assumption all tangent spaces should have the same dimension and we want to approximate $\hat{M}$ with a set of matrices $\hat{X}$ where each X_i has a low rank. This can be formulated as

$$\min_{\hat{X}} \mu r(\hat{X}) + \|\hat{X} - \hat{M}\|^2. \tag{6.41}$$

The task is thus to find a low-rank approximation X_i of each M_i. Further, since we are interested in the affine tangent spaces, which do not necessarily go through the origin, the row-wise mean vector $\bar{\mathbf{x}}_i$ is added to each X_i. Together with the assumption that $X_i \mathbb{1} = 0$, the fitting terms in the objective can be written

$$\sum_{i=1}^{b} \|X_i + \bar{\mathbf{x}}\mathbb{1}^T - M_i\|_F^2 = \tag{6.42}$$

$$\sum_{i=1}^{b} \|X_i - (M_i - \bar{\mathbf{m}}_i\mathbb{1}^T)\|_F^2 + k_i\|\bar{\mathbf{x}}_i - \bar{\mathbf{m}}_i\|_2^2, \tag{6.43}$$

where $\bar{\mathbf{m}}_i$ is the row-mean of M_i and k_i is the number of columns in M_i. To ensure consistency between shared variables, the difference is penalized by adding the term

$$\alpha \sum_{i=1}^{b} \|\mathcal{P}_i(X) - (X_i + \bar{\mathbf{x}}_i\mathbb{1}^T)\|^2, \tag{6.44}$$

where α is a weighting factor, X is the approximation of the measurement matrix M and $\mathcal{P}_i(X)$ is a linear operator that retrieves block i in X. This results in the objective

$$\begin{aligned} \min_{\hat{X},X,\hat{\bar{\mathbf{x}}}} r(\hat{X}) + \sum_{i=1}^{b}(\|X_i - (M_i - \bar{\mathbf{m}}\mathbb{1}^T)\|_F^2 + \\ k_i\|\bar{\mathbf{x}}_i - \bar{\mathbf{m}}_i\|_2^2 + \alpha\|\mathcal{P}_i(X) - (X_i + \bar{\mathbf{x}}_i\mathbb{1}^T)\|_F^2) \quad (6.45) \\ \text{s.t.}\, X_i\mathbb{1}^T = 0, \quad i = 1, \ldots, b. \quad (6.46) \end{aligned}$$

For the terms in the first row in (6.45) the convex envelope has already been derived. In the second row the terms are already convex. Also the constraint (6.46) is linear. To minimize this we use the convex envelope and add the auxiliary variable Z_i. This gives

$$\min_{\hat{X},X,\hat{\bar{\mathbf{x}}},\hat{Z}} f^{**}(\hat{X}) + \sum_{i=1}^{b}(k_i\|\bar{\mathbf{x}}_i - \bar{\mathbf{m}}_i\|_2^2 + \alpha\|\mathcal{P}_i(X) - Z_i - \bar{\mathbf{x}}_i\mathbb{1}^T\|_F^2) \quad (6.47)$$

$$\text{s.t.} \quad X_i = Z_i, \quad (6.48)$$

$$Z_i\mathbb{1}^T = 0, \quad i = 1,\dots,b, \quad (6.49)$$

which in turn, leads to the ADDM formulation [11]

$$\min_{\hat{X},X,\hat{\bar{\mathbf{x}}},\hat{Z}} f^{**}(\hat{X}) + \rho\|\hat{X} - \hat{Z} + \hat{\Lambda}\|^2 - \rho\|\hat{\Lambda}\|^2 +$$

$$\sum_{i=1}^{b}(k_i\|\bar{\mathbf{x}}_i - \bar{\mathbf{m}}_i\|_2^2 + \alpha\|\mathcal{P}_i(X) - Z_i - \bar{\mathbf{x}}_i\mathbb{1}^T\|_F^2) \quad (6.50)$$

$$\text{s.t. } Z_i\mathbb{1}^T = 0, \quad i = 1,\dots,b. \quad (6.51)$$

Here one part depends on $\hat{X}$,

$$\min_{\hat{X}} f^{**}(\hat{X}) + \rho\|\hat{X} - \hat{Z} + \hat{\Lambda}\|_F^2 \quad (6.52)$$

which is precisely the proximal operator in (6.21) which we know how to minimize. Minimizing the other variables X, $\hat{Z}$ and $\hat{\bar{\mathbf{x}}}$ is now straightforward. Keeping all other variables fixed and solving for one we get the updates:

$$X^{t+1} = \arg\min_{X^t} \alpha\sum_{i=1}^{b}\|\mathcal{P}_i(X^t) - Z_i^t - \bar{\mathbf{x}}_i^t\mathbb{1}^T\|_F^2, \quad (6.53)$$

which is a separable least square problem.

$$Z_i^{t+1} = \arg\min_{Z_i^t} \alpha\|\mathcal{P}_i(X^{t+1}) - Z_i^t - \bar{\mathbf{x}}_i^t\mathbb{1}^T\|^2 + \quad (6.54)$$

$$\rho\|X_i^t - Z_i^t + \Lambda_i^t\|_F^2, \ i = 1,\dots,b. \quad (6.55)$$

since all blocks are independent in the minimization,

$$\bar{\mathbf{x}}_i^{t+1} = \arg\min_{\bar{\mathbf{x}}_i^t} k_i\|\bar{\mathbf{x}}_i^t - \bar{\mathbf{m}}_i\|_2^2 + \alpha\|\mathcal{P}_i(X^{t+1}) - Z_i^{t+1} - \bar{\mathbf{x}}^t\mathbb{1}^t\|_F^2, \quad (6.56)$$

	Input PSNR	Output PSNR
Our method	10.46	**17.52**
Manifold Denoising	10.46	15.67

Table 6.1: The PSNR using different methods for denoising on the USPS Digits.

since we can minimize each $\bar{\mathbf{x}}_i$ separately. Finally, $\hat{X}^{t+1}$ is found by using the proximal operator

$$\hat{X}^{t+1} = \text{prox}_{f^{**}}(\hat{Z}^{t+1} - \hat{\Lambda}^t). \tag{6.57}$$

$\hat{\Lambda}$ is updated by taking a step in the ascent direction

$$\Lambda_i^{t+1} = \Lambda_i^T + X_i^{t+1} - Z_i^{t+1}, \tag{6.58}$$

since again, the blocks are independent.

Manifold Denoising - Experimental Results

Our method is tested on the USPS dataset [36] of handwritten digits. 100 images of each digit are selected and the intensity is rescaled to lie between $[0, 1]$. The images are then perturbed with Gaussian noise with standard deviation $\sigma = 0.3$. Thereafter *all* images are stacked into one measurement matrix M and for each column (image) the K closest neighbors are found. In this experiment $K = 30$. Our method is then applied to this data to obtain an approximation X of M. For comparison, the work [33] is applied to the data as well. This paper takes a different approach by creating a graph of the data and solving partial differential equations on this graph. The result of our experiment is shown in Table 6.1. The parameters used in [33] was $\lambda = 1$, number of neighbors was 6 and a symmetric graph was used since that gave the best results on this data. Some of the data and denoised images are shown in Figure 6.3.

Manifold Denosing - Single Image Denoising

In the single image case, there is only one image of the object. However, if the image is subdivided into patches, then several of these patches should be similar and the same idea can be applied. The denoised patches are then used to recreate the image. As in the USPS dataset, all patches are vectorized and put into a measurement matrix M and we find the optimal X by applying our method.

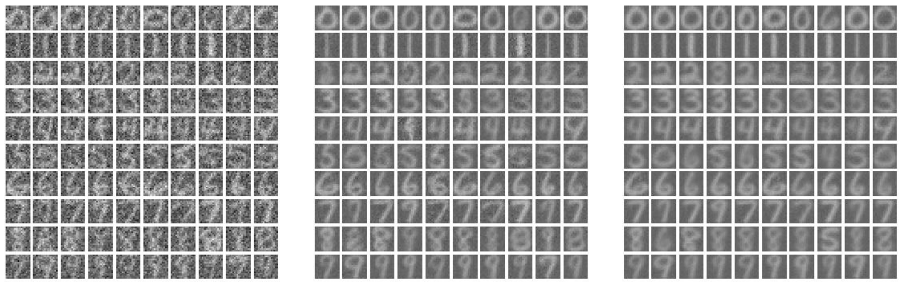
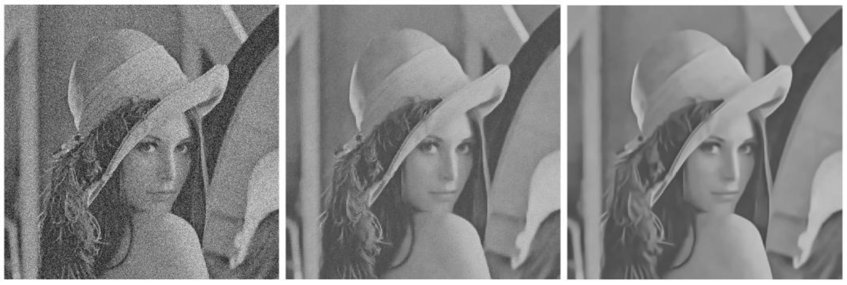
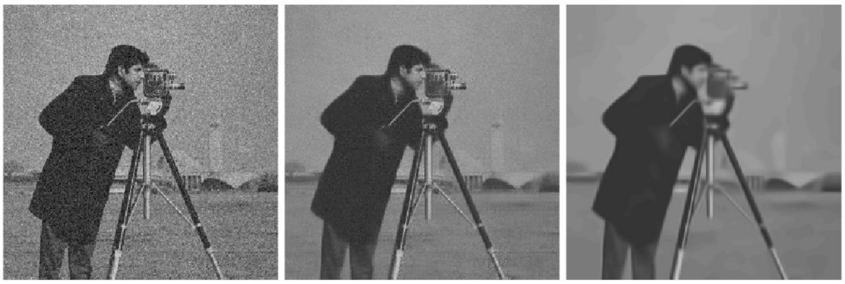


Figure 6.3: The result from the USPS dataset. *Left:* Corrupted images with Gaussian noise of standard deviation $\sigma = 0.3$. *Middle:* Our Results. *Right:* Results from Manifold Denoising [33].



(a) Input image (b) Denoised image our method (c) Denoised with BM3D

Figure 6.4: *From left to right*: Noisy input image, denoised image with our method and denoised image with BM3D [23].



(a) Noisy input image (b) Denoised with our method (c) Denoised with BM3D.

Figure 6.5: Result from experiment on the cameraman images.

	Input	Our method	BM3D
Lena	19.99	28.61	**29.16**
Cameraman	19.99	**26.07**	24.73

Table 6.2: Denoising results from the Cameraman and Lena. BM3D gives a higher PSNR for Lena, but we do better on the Cameraman.

This idea was tested on two images, *Lena* of size 512×512 and the *Cameraman* of size 256×256. Gaussian noise with a standard deviation of $\sigma = 0.1$. As can be seen in Figure 6.4 the noise is reduced. We also provide results from the state-of-the-art method BM3D [23]. If one looks at the close-up in Figure 6.6 the details are more distinguishable in our result. For example the eye is more detailed. These results were obtained by using a patch size of 12×12 pixels and with an overlap of 2/3 between two consecutive patches, resulting in 15876 patches. The parameter α was set to 1.5, $\mu = 75000$ and the number of neighbors $K = 20$. The optimal blocks X_i had rank 2.

The same approach was also tested on the Cameraman and as can be seen in Figure 6.5, our method performs well compared to BM3D which smooths out some details. For example the face and the camera is more detailed in our denoised image. In this experiment the same parameters as above was used, except for $\mu = 22000$ and the number of patches was 3844. The rank of the optimal blocks X_i was 3. In Table 6.2 we show quantitative results from the experiments. It can be seen that our method is comparable to BM3D on this data.

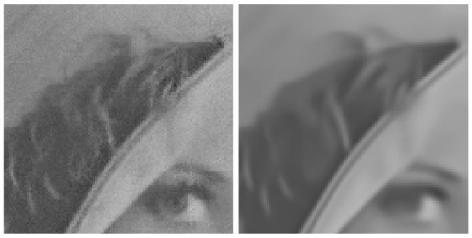


Figure 6.6: Zooming in, one can see more details in our result (*left*) compared to BM3D's result (*right*). Note that one can see the pupil in the eye of the left image, but not in the right image.

Dataset	Loc. Rank Func. [45]	Our method
Hand	0.474	0.474
Banner	$6.54 \cdot 10^7$	$4.73 \cdot 10^7$
Book	0.121	0.121

Table 6.3: The error $\sum_{i=1}^{b} \|X_i - M_i\|_F^2$ for the method in Section 5.2 and our method. Note that the method in Section 5.2 outperforms the nuclear norm relaxation for the same error metric.

6.3.2 Linear Shape Basis Estimation

Another application we test our method on is estimation of linear shape models. A common assumption is that a set of tracked image points moving non-rigidly can be described with a small number of basis elements in each frame. In this experiment we let M^f denote the N tracked 2D points in frame f, we want to find a shape basis model $(\mathbf{s}_1, \dots, \mathbf{s}_K)$, each of size $2 \times N$ and scalar coefficients $(c_1^f, \dots, c_K^f)$ so that the points M^f can be written

$$M^f = \sum_{k=1}^{K} c_k^f \mathbf{s}_k. \tag{6.59}$$

Stacking N points in F frames yields a $2F \times N$ measurement matrix M. The fewer basis elements we have, the simpler our model is, so we want M to have low rank.

In contrast to the image denoising experiments, there will be missing data here due to occlusion and tracking failures. As in Section 5.2, this is handled by dividing M into sub-blocks M_i with no missing data, see right of 6.2. The problem is now to find low-rank approximations of the sub-blocks X_i such that they coincide on the overlap. The objective function to be minimized is then

$$\min_{\hat{X},X)} f^{**}(\hat{X}) \tag{6.60}$$

$$\text{s.t.} \quad \mathcal{P}_i(X) = X_i, \quad i = 1, \dots, b. \tag{6.61}$$

Again $\hat{X}$ is the collection of blocks $(X_1, \dots, X_b)$ and $\mathcal{P}_i(X)$ retrieves block i from X. The constraint (6.61) ensures that the overlap between the blocks coincide. As optimization method ADMM is used and our augmented Lagrangian becomes

$$f^{**}(\hat{X}) + \rho\|\hat{X} - \hat{\mathcal{P}}(X) + \hat{\Lambda}\|^2 - \rho\|\hat{\Lambda}\|^2, \tag{6.62}$$

where $\hat{\mathcal{P}}(X) = (\mathcal{P}_1(X), \dots, \mathcal{P}_b(X))$. The updates performed in each iteration are

$$\hat{X}^{t+1} = \arg\min_{\hat{X}^t} f^{**}(\hat{X}^t) + \rho\|\hat{X}^t - \hat{\mathcal{P}}(X^t) + \hat{\Lambda}^t\|^2 \tag{6.63}$$

$$X^{t+1} = \arg\min_{X^t} \rho\|\hat{X}^{t+1} - \hat{\mathcal{P}} + \hat{\Lambda}^t\|^2 \tag{6.64}$$

$$\Lambda_i^{t+1} = \Lambda_i^t + X_i^{t+1} - \mathcal{P}_i(X^{t+1}). \tag{6.65}$$

After optimization, we complete the missing parts in X using the same procedure as in Section 5.2.

Linear Shape Basis - Experimental results

Our method is applied to the same image sequences as before, namely *Hand*, *Book* and *Banner*. The results from the experiment are shown in Figures 6.7, 6.8 and 6.9. In all sequences the red and blue points that are reconstructed obey a reasonable motion compared to input data, green points. The blue points are reconstructed positions that could be tracked and red points are reconstructed positions where we had no measurements. The obtained rank for the solution in the *Hand* sequence is 5, in the book sequence we get 3 and the banner sequence we get rank 9. The number of blocks in *Hand*, *Book* and *Banner* were 5, 3 and 19.

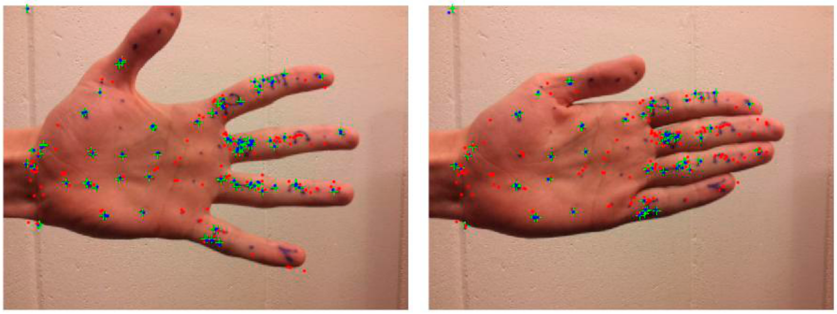


Figure 6.7: Frames 280 and 371 from the hand experiment. The solution has rank 5.

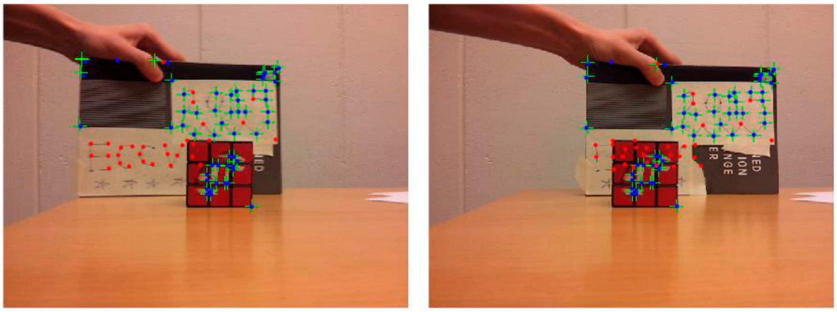


Figure 6.8: Frames 297 and 337 in the Book sequence. The solution has rank 3.

To compare to the method in Section 5.2, we test the method on the same datasets and measure the error $\sum_{i=1}^{b} \|M_i - X_i\|_F^2$ and the results are shown in Table 6.3. We choose to measure the error on the blocks since that will show if our new method differs from the old one.

As the results implies, our method performs equally well as our method in Section 5.2 on these datasets. The *Banner*-sequence differs to our advantage. Looking closer on the resulting sub-blocks from our other method shows that some blocks have rank 8 and some rank 10. This suggests that it is easier to get a uniform rank with our new method.

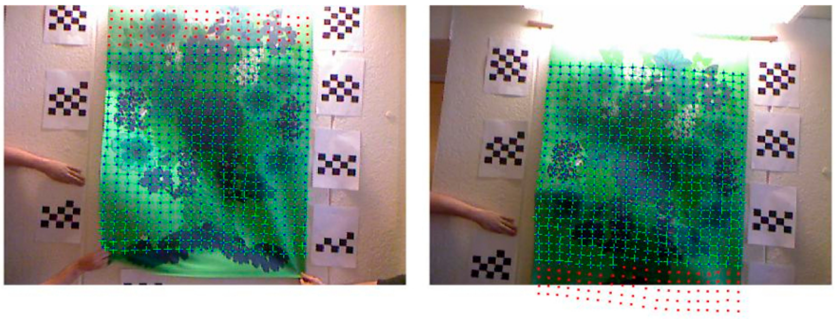


Figure 6.9: Frames 160 and 250 of the Banner sequence. The solution has rank 9.

6.4 Conclusion

In this chapter we saw that we could derive an objective function that simultaneously penalizes the rank of all matrices in the objective function. A convex envelope was derived which could be combined with other constraints for different applications. This is a natural formulation in applications like denoising where one want all matrices to have the same rank. It was shown in experiments that our method performed well compared to specialized denoising methods like BM3D [23].

Chapter 7

Under Determined Linear Systems and Rank Regularization

7.1 Introduction

In the two previous chapters, we saw two different convex envelopes of the problem

$$\min_{X} \; \mu g_r(X) + \|X - M\|_F^2, \tag{7.1}$$

where $g_r(X)$ is a function that penalizes the rank of the solution. The derived convex envelopes have the advantage compared to the nuclear norm [58] that they do not penalize large singular values. Here we study the problem

$$\min_{X} \; \mathbb{I}(\text{rank}(X) \leq r) + \|\mathcal{A}X - \mathbf{b}\|^2, \tag{7.2}$$

where $\mathbb{I}(\text{rank}(X) \leq r)$ is 0 if $\text{rank}(X) \leq r$ and ∞ else. The linear operator $\mathcal{A} : \mathbb{R}^{m \times n} \to \mathbb{R}^p$ is assumed to fulfill a restricted isometry property (RIP) [58]

$$(1 - \delta_q)\|X\|_F^2 \leq \|\mathcal{A}X\|^2 \leq (1 + \delta_q)\|X\|_F^2, \tag{7.3}$$

for all matrices with $\text{rank}(X) \leq q$.

This type of objective has applications in for example non-rigid structure from motion, see [25], where the linear operator is the matrix of all rotations. There are also applications in control theory that are for example studied in [58], for instance, estimating a Hankel matrix can be cast into solving

$$\min_{h} \text{rank}(\text{hankel}(h))) + \|Ah - b\|^2. \tag{7.4}$$

The common approach to tackle this type of problem is to again replace the rank function with the nuclear norm [58, 55]. Performance guarantees for replacing the rank-function with the nuclear norm appear in [18, 58, 55, 17]. Still the approach suffers from the shrinking bias induced by the nuclear norm. Some other approaches [49, 56] propose non-convex alternatives which have shown improved performance.

Here we consider the relaxation

$$\min_X \; \mathcal{R}_r(X) + \|\mathcal{A}X - \mathbf{b}\|^2, \tag{7.5}$$

where

$$\mathcal{R}_r(X) = \max_Z \sum_{i=r+1}^{N} z_i^2 - \|X - Z\|_F^2, \tag{7.6}$$

and z_i, $i = 1, \dots, N$ are the singular values of Z. It was shown in [44] that

$$\mathcal{R}_r(X) + \|X - X_0\|_F^2 \tag{7.7}$$

is the convex envelope of

$$\mathbb{I}(\text{rank}(x) \leq r) + \|X - X_0\|_F^2. \tag{7.8}$$

By itself the regularization term $\mathcal{R}_r(X)$ is not convex, but when adding a quadratic term $\|X\|_F^2$ the result is convex. If the singular values of X_0 are distinct, it was shown in [3] that the minimizers to (7.7) and (7.8) are the same.

If we assume that (7.3) holds, then $\|\mathcal{A}X\|^2$ behaves roughly like $\|X\|_F^2$ if the matrix has rank less than q. Therefore the idea is that the problem (7.5) should have some convexity properties. Here we will study the stationary points of (7.5) and we will see that if the RIP-property holds, then it is possible in many cases to guarantee that any stationary point of rank r is unique.

7.2 Main Results and Contributions

Our main contribution is that we show that if X_s is a stationary point of (7.5) and the singular values of the matrix

$$Z = (I - \mathcal{A}^*\mathcal{A})X_s + \mathcal{A}^*b \tag{7.9}$$

fulfill $z_{r+1} < (1 - 2\delta_{2r})z_r$ then there cannot be any other stationary point with rank less than or equal to r. For example, if there is a rank r matrix X_0 such that $\mathcal{A}X_0 = \mathbf{b}$ then it is easy to show that X_0 is a stationary point and the corresponding Z is identical to X_0. This means that if $z_{r+1} = 0$ our results certify that this is the only rank r stationary point to the problem if $\mathcal{A}$ fulfills (7.3) with $\delta_{2r} < \frac{1}{2}$.

To see the connection between the stationary point X_s and Z we have the following lemma:

Lemma 7.2.1. The point X_s is stationary in $F(X) = \mathcal{R}_r(X) + \|\mathcal{A}X - \mathbf{b}\|^2$ if and only if $2Z \in \partial G(X_s)$, where $G(X) = \mathcal{R}_r(X) + \|X\|_F^2$ and if and only if

$$X_s \in \arg\min_X \mathcal{R}_r(X) + \|X - Z\|_F^2. \tag{7.10}$$

Proof. We can write $F(X) = G(X) - \delta_q\|X\|_F^2 + H(X) + \|\mathbf{b}\|^2$ with $H(X) = \delta_q\|X\|_F^2 + (\|\mathcal{A}X\|^2 - \|X\|_F^2) - 2\langle \mathcal{A}X, \mathbf{b}\rangle$. Taking the sub-differential of G and the derivative of the remaining terms yields

$$2\delta_q X_s - \nabla H(X_s) \in \partial G(X_s). \tag{7.11}$$

We have $\nabla H = 2\delta_q X + 2(\mathcal{A}^*\mathcal{A} - I)X - 2\mathcal{A}^*\mathbf{b}$ and inserting this above we get

$$2(I - \mathcal{A}^*\mathcal{A})X_s + 2\mathcal{A}^*\mathbf{b} = 2Z \in \partial G(X_s). \tag{7.12}$$

Hence, X_s is stationary if and only if $2Z \in \partial G(X_s)$. If we put $\mathcal{A} = \mathcal{I} : \mathbb{R}^{m\times n} \to \mathbb{R}^{mn}$ and $\mathbf{b} = \hat{Z}$ which is Z vectorized, then we have (7.10) and we get that X_s is stationary in (7.10) if and only if $2Z \in \partial G(X_s)$. Since (7.10) is convex, getting the stationary point is equivalent to finding the minimizer.

□

In [3] it is shown that as long as $z_r \neq z_{r+1}$ the unique solution of (7.10) is the best rank r approximation of Z. When there are several singular values that are equal to z_r, (7.10) will have multiple solutions where some are of rank $\neq r$.

This work builds on that of [52] where a similar result is derived for the non-convex regularizer $\mathcal{R}_\mu(X) = \sum_i \mu - \max(\sqrt{\mu} - x_i, 0)^2$. In this case the trade-off between rank and residual error is optimized using the formulation

$$\min_X \mathcal{R}_\mu(X) + \|\mathcal{A}X - \mathbf{b}\|^2. \tag{7.13}$$

Since it is possible to select μ such that (7.13) gives the desired rank, it is possible to argue that (7.13) is practically equivalent to (7.1). However, it is shown in [20] that if $\|\mathcal{A}\| \leq 1$, then a local minimizer of (7.5) is a local minimizer of (7.1) and their global minimizers coincide. In particular, if $\|\mathcal{A}\| \leq 1$, with our result, a local minimizer X_s of (7.5) will have rank lower than or equal to r and will be unique. In contrast, [52] does not rule out the existence of multiple high rank solutions.

7.2.1 Subgradients of G

We now need to determine the subdifferential $\partial G(X)$ of the function $G(X)$. Let $\mathbf{x}$ be the vector of singular values of X and $X = UD_{\mathbf{x}}V^T$ be the SVD. From von Neumann's trace theorem it is easy to see that the Z that maximizes (7.6) has the same U and V as X and is on the form $Z = UD_{\mathbf{z}}V^T$, where $\mathbf{z}$ is the vector of singular values for Z.

If we let

$$L(X, Z) = -\sum_{i=1}^{r} z_i^2 + 2\langle Z, X\rangle, \tag{7.14}$$

then we get that $G(X) = \max_Z L(X, Z)$. The function L is concave in Z and linear in X. Due to the dominating quadratic term, the domain can be restricted to a compact set. From Danskin's theorem, see [8], we get that the subgradients of G are then given by

$$\partial G(X) = \text{convhull}\{\nabla_X L(X, Z), Z \in \mathcal{Z}(X)\}, \tag{7.15}$$

where $\mathcal{Z}(X) = \arg\max_Z L(X, Z)$. Further, by concavity, the set $\mathcal{Z}$ is convex. Since $\nabla_X L(X, Z) = 2Z$ we get

$$\partial G(X) = 2 \arg\max_Z L(X, Z). \tag{7.16}$$

In order to find the set of subgradients we have to determine all maximizers of L. Since the maximizing Z has the same U and V as X, it remains to find the singular values of Z. It can be shown, see [44], that these have the form

$$z_i \in \begin{cases} \max(x_i, s) & \text{if } i \leq r \\ s & \text{if } i \geq r,\, x_i \neq 0, \\ [0, s] & \text{if } i > r,\, x_i = 0 \end{cases} \tag{7.17}$$

for some number $s \geq x_r$. If rank$(X) \leq r$ the optimal choice is $s = x_r$. This is seen by observing that since the optimal Z is of the form UD_zV^T we have

$$L(X, Z) = -\sum_{i=1}^{r}(z_i - x_i)^2 + \sum_{i=1}^{r} x_i^2, \tag{7.18}$$

since rank$(X) \leq r$. Selecting $s = x_r$ and inserting this into (7.17) gives $L(X, Z) = \sum_{i=1}^{r} x_i^2$ which is clearly the optimum. If rank$(X) > r$, then a one dimensional concave and differentiable function needs to be maximized numerically [44].

7.3 Growth Estimates for the $\partial G(X)$

The next step is to estimate the growth of the subgradients that is needed when the uniqueness of low rank stationary points is considered. Let $\mathbf{x}$ and $\mathbf{x}'$ be two vectors with at most r non-zero positive elements and I and I' be the indexes of the r largest elements of $\mathbf{x}$ and $\mathbf{x}'$ respectively. It is assumed that both I and I' contain r elements. If in particular $\mathbf{x}'$ has fewer than r non-zero elements, some zero elements are included in I'. We define the corresponding sequences $\mathbf{z}$ and $\mathbf{z}'$ by

$$z_i \in \begin{cases} x_i & i \in I \\ [0, \bar{s}] & i \notin I, \end{cases} \tag{7.19}$$

$$z_i' \in \begin{cases} x_i' & i \in I' \\ [0, \bar{s}'] & i \notin I', \end{cases} \tag{7.20}$$

where $\bar{s} = \min_{i \in I} x_i$ and $\bar{s}' = \min_{i \in I'} x_i'$. If $\mathbf{x}'$ has fewer than r non-zero elements then $\bar{s}' = 0$. Note that we do not require that the elements in $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{z}$ and $\mathbf{z}'$ are ordered in decreasing order. For the following lemma denote $\underline{s} = \max_{i \notin I} z_i$.

Lemma 7.3.1. If $\underline{s} < c\bar{s}$, where $0 < c < 1$ then

$$\langle \mathbf{z}' - \mathbf{z}, \mathbf{x}' - \mathbf{x} \rangle > \frac{1-c}{2} \|\mathbf{x}' - \mathbf{x}\|^2. \tag{7.21}$$

Proof. Since $z_i = x_i$ when $i \in I$ and $x_i = 0$ otherwise, we can write the inner product $\langle \mathbf{z}' - \mathbf{z}, \mathbf{x}' - \mathbf{x} \rangle$ as

$$\sum_{\substack{i\in I\\ i\in I'}} (x_i - x_i')^2 + \sum_{\substack{i\in I\\ i\notin I'}} x_i(x_i - z_i') + \sum_{\substack{i\notin I\\ i\in I'}} x_i'(x_i' - z_i). \tag{7.22}$$

We also have

$$\|\mathbf{x}' - \mathbf{x}\|^2 = \sum_{\substack{i\in I\\ i\in I'}} (x_i - x_i')^2 + \sum_{\substack{i\in I\\ i\notin I'}} x_i^2 + \sum_{\substack{i\notin I\\ i\in I'}} x_i'^2. \tag{7.23}$$

Due to our initial assumption that I and I' have the same number of elements, the second and third term in (7.23) will have the same number of terms. Therefore if we can show that

$$x_i(x_i - z_i') + x_j'(x_j' - z_j) \geq \frac{1-c}{2}(x_i^2 + x_j'^2) \tag{7.24}$$

when $i \in I$, $i \notin I'$ and $j \notin I$, $j \notin I'$, we have showed the wanted inequality. By the assumption $\underline{s} < c\bar{s}$ we know that $z_j < cx_i$. We further know that

$$z_i' \leq \bar{s}' \leq x_j'. \tag{7.25}$$

This gives

$$x_i z_i' \leq x_i x_j' \leq \frac{x_i^2 + x_j'^2}{2} \tag{7.26}$$

$$x_j' z_j < c x_j' x_i \leq c\frac{x_i^2 + x_j'^2}{2} \tag{7.27}$$

Inserting these inequalities into the left hand side of (7.24) gives the desired bound. The following two results are from [52] and are needed to prove the main theorem. The proofs are included for completeness.

Lemma 7.3.2. Let $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{z}$ and $\mathbf{z}'$ be fixed vectors with non-increasing and non-negative elements such that $\mathbf{x} \neq \mathbf{x}'$ and $\mathbf{z}$, $\mathbf{z}'$ fulfill (7.17). Define $X' = U'D_{\mathbf{x}'}V'^T$, $X = UD_{\mathbf{x}}V^T$, $Z' = U'D_{\mathbf{z}'}V'^T$ and $Z = UD_{\mathbf{z}}V^T$ as functions of unknown orthogonal matrices U, V, U' and V'. If

$$a^* = \min_{U,V,U',V'} \frac{\langle Z' - Z, X' - X \rangle}{\|X - X'\|_F^2} \leq 1 \tag{7.28}$$

then

$$a^* = \min_{M_\pi} \frac{\langle M_\pi \mathbf{z}' - \mathbf{z}, M_\pi \mathbf{x}' - \mathbf{x} \rangle}{\|M_\pi \mathbf{x}' - \mathbf{x}\|^2}, \tag{7.29}$$

where M_π is a permutation matrix.

Proof. We may assume that $U = I_{m\times m}$ and $V = I_{n\times n}$. We first note that (U', V') is a minimizer of (7.28) if and only if

$$\langle Z' - Z, X' - X \rangle \leq a^* \|X' - X\|_F^2. \tag{7.30}$$

This constraint can equivalently be written

$$C - \langle Z' - a^* X', X \rangle - \langle Z - a^* X, X' \rangle \leq 0, \tag{7.31}$$

where $C = \langle Z', X' \rangle + \langle Z, X \rangle - a^*(\|X'\|_F^2 + \|X\|_F^2)$ is independent of U' and V'. Thus any minimizer of (7.28) must also maximize

$$\langle U' D_{\mathbf{z}' - a^* \mathbf{x}'} V'^T, D_x \rangle + \langle D_{\mathbf{z} - a^* \mathbf{x}}, U' D_{\mathbf{x}'} V'^T \rangle. \tag{7.32}$$

For ease of notation we now assume that $m \leq n$, that is, the number of rows are less than the columns (the opposite case will be handled by transposing). Equation (7.32) can now be written

$$\mathbf{x}^T M(\mathbf{z}' - a^* \mathbf{x}') + (\mathbf{z} - a^* \mathbf{x})^T M \mathbf{x}', \tag{7.33}$$

where $M = U' \odot V'_{1,1}$, $V'_{1,1}$ is the upper left $m \times m$ block of V' and $\odot$ denotes element wise multiplication. Since both U' and V' are orthogonal it is easily shown by using the Cauchy-Schwartz inequality that M is DSS, a doubly substochastic matrix.

Note that objective (7.32) is linear in M and therefore optimization over the set of DSS matrices is guaranteed to have an extreme point $M_{\pi,\mathbf{v}}$ that is optimal. Furthermore, since $a^* \leq 1$ the vectors $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{z} - a^*\mathbf{x}$ and $\mathbf{z}' - a^*\mathbf{x}'$ all have positive entries and therefore the maximizing matrix has to be $M_{\pi,\mathbb{1}}$ for some permutation π. Since $M_{\pi,\mathbb{1}}$ is orthogonal and $M_{\pi,\mathbb{1}} = M_{\pi,\mathbb{1}} \odot M_{\pi,\mathbb{1}}$, $U' = M_{\pi,\mathbb{1}}$ and $V'_{1,1} = M_{\pi,\mathbb{1}}$ will be optimal when maximizing (7.33) over U' and $V'_{1,1}$. An optimal V' in (7.32) can now be chosen to be $V' = \begin{bmatrix} M_{\pi,\mathbb{1}} & 0 \\ 0 & I \end{bmatrix}$. Note this choice is somewhat arbitrary since only the upper left block of V' affects the value of (7.32). The matrices $U'Z'V'^T$ and $U'X'V'^T$ are now diagonal, with diagonal elements $M_{\pi,\mathbb{1}}\mathbf{z}'$ and $M_{\pi,\mathbb{1}}\mathbf{x}'$, which concludes the proof.

□

Corollary 7.3.3. Assume X is of rank r and $2Z \in \partial G(X)$. If the singular values of the matrix Z fulfill $z_{r+1} < cz_r$, where $0 < c < 1$, then for any $2Z' \in \partial G(X')$ with $\text{rank}(X') \leq r$ we have

$$\langle Z' - Z, X' - X \rangle > \frac{1-c}{2} \|X' - X\|_F^2. \tag{7.34}$$

as long as $\|X' - X\|_F^2 \neq 0$.

Proof. We let $\mathbf{x}$, $\mathbf{x}'$, $\mathbf{z}$ and $\mathbf{z}'$ be the singular values of the matrices X, X', Z and Z' respectively. Our proof essentially follows that of Corollary 4.2 in [52], where a similar result is proven under the assumption that $\mathbf{x} \neq \mathbf{x}'$ and then generalized to the general case using a continuity argument. To use the continuity argument, the infeasible region must be extended somewhat. Since $0 < c < 1$ and $z_{r+1} < cz_r$ are open there is an $\epsilon > 0$ such that $z_{r+1} < (c - \epsilon)z_r$ and $0 < c - \epsilon < 1$. Now assume $a^* > 1$ in (7.28), then clearly

$$\langle Z' - Z, X' - X \rangle > \frac{1-(c-\epsilon)}{2} \|X' - X\|_F^2. \tag{7.35}$$

Otherwise $a^* \leq 1$ and we have

$$\frac{\langle Z' - Z, X' - X \rangle}{\|X' - X\|_F^2} \geq \frac{\langle M_\pi \mathbf{z}' - \mathbf{z}, M_\pi \mathbf{x}' - \mathbf{x} \rangle}{\|M_\pi \mathbf{x}' - \mathbf{x}\|^2}. \tag{7.36}$$

According to Lemma 7.3.1 the right hand side is strictly larger than $\frac{1-(c-\epsilon)}{2}$ which proves that (7.35) for all X' with $\mathbf{x}' \neq \mathbf{x}$.

It remains to show that

$$\langle Z' - Z, X' - X \rangle \geq \frac{1-(c-\epsilon)}{2} \|X' - X\|_F^2, \tag{7.37}$$

in the case $\mathbf{x}' = \mathbf{x}$ and $\|X' - X\|_F^2 \neq 0$. For that define the sequence

$$\sigma_1(t) = \sigma_1(X) + t \tag{7.38}$$

$$\sigma_i(t) = \sigma_i(X), \forall i \neq 1 \tag{7.39}$$

$$X(t) = UD_{\sigma(t)}V^T. \tag{7.40}$$

For all $t > 0$, the vectors $\mathbf{x}' \neq \mathbf{x}(t)$ and then (7.35) holds. By continuity of the Frobenius norm and the scalar product, (7.37) is true if we let $t \to 0$.

□

7.3.1 Uniqueness of Low Rank Stationary Points

We can now show that if a RIP-inequality (7.3) holds and the singular values are well separated there can only be one stationary point of F that has rank r. For that we first derive a bound on the gradients of H. We have

$$\nabla H(X) = 2\delta_q X + 2(\mathcal{A}^*\mathcal{A} - I)X - 2\mathcal{A}^*\mathbf{b}. \tag{7.41}$$

This gives

$$\begin{aligned} \langle \nabla H(X') - \nabla H(X), X' - X\rangle = \\ 2\delta_q \|X' - X\|_F^2 + 2(\|\mathcal{A}(X' - X)\|^2 - \|X' - X\|_F^2). \end{aligned} \tag{7.42}$$

From the RIP-property we also have if $\text{rank}(X - X') \leq q$

$$|\|X - X'\|_F^2 - \|\mathcal{A}(X - X')\|^2| \leq \delta_q \|X - X'\|_F^2 \tag{7.43}$$

which gives

$$\langle \nabla H(X') - \nabla H(X), X' - X\rangle \geq 0. \tag{7.44}$$

With these observations we can now state the main result:

Theorem 7.3.4. Assume that X_s is a stationary point of F, that is $(I - \mathcal{A}^*\mathcal{A})X_s + \mathcal{A}^*\mathbf{b} = Z$, where $2Z \in \partial G(X_s)$, $\text{rank}(X_s) = r$ and the singular values of Z fulfill $z_{r+1} < (1 - 2\delta_{2r})z_r$. If X_s' is another stationary point then $\text{rank}(X_s') > r$.

Proof. Assume that $\text{rank}(X_s') \leq r$. Since both X_s and X_s' are stationary we have

$$2\delta_{2r} X_s' - \nabla H(X_s') = 2Z', \tag{7.45}$$

$$2\delta_{2r} X_s - \nabla H(X_s) = 2Z, \tag{7.46}$$

where $2Z \in \partial G(X_s)$ and $2Z' \in \partial G(X_s')$. Taking the difference between the two equations yields

$$2\delta_{2r}(X_s' - X_s) - \nabla H(X_s) + \nabla H(X_s) = 2Z' - 2Z, \tag{7.47}$$

which implies

$$2\delta_{2r}\|V\|_F^2 - \langle \nabla H(X_s') + \nabla H(X_s), V\rangle = 2\langle Z' - Z, V\rangle \tag{7.48}$$

where $V = X_s' - X_s$ has $\text{rank}(V) \leq 2r$. By (7.44) the left hand side is less than $\delta_{2r}\|V\|_F^2$. On the other hand, Corollary 7.3.3 gives with $c = 1 - \delta_{2r}$ that the right hand side is larger than $2\partial_{2r}\|V\|_F^2$ which contradicts $\text{rank}(X_s') \leq r$.

7.3.2 Implementation and Experiments

In this section the proposed method is tested on some simple real and synthetic applications, where some fulfill (7.3) and some do not. To optimize the objective the GIST approach is used from [31]. Given a current iterate X_k this method solves

$$X_{k+1} = \arg\min_X \mathcal{R}_r(X) + \tau_k \|X - M_k\|_F^2, \tag{7.49}$$

where $M_k = X_k - \frac{1}{\tau_k}(\mathcal{A}^*\mathcal{A}X_k - \mathcal{A}^*\mathbf{b})$ From Lemma 7.2.1 we see that for $\tau_k = 1$, any fixed point of (7.49) is a stationary point. To solve (7.49) we use the proximal operator computed in [44].

We optimize (7.49) for a sequence of $\{\tau_k\}$. As initialization $\tau_0 = 5$ and then τ_k is reduced toward 1. If the objective value is successfully decreased in the previous step, we update τ_k by $\tau_{k+1} = \frac{\tau_k - 1}{1.1} + 1$, if the objective is not decreased then we update τ_{k+1} by $\tau_{k+1} = 1.5(\tau_k - 1) + 1$.

7.3.3 Synthetic Data

We first evaluate the quality of the relaxation on a number of synthetic experiments. We compare the two formulas (7.5) and

$$\min_X \mu\|X\|_* + \|\mathcal{A}X - \mathbf{b}\|^2. \tag{7.50}$$

In Figure 7.1 we tested these two relaxations on a number of synthetic problems with varying noise level. The data was created so that the operator $\mathcal{A}$ fulfills (7.3) with $\delta = 0.2$. By column stacking the $m \times n$ matrix X the linear mapping $\mathcal{A}$ can be represented with a matrix A of size $p \times mn$. For the data in Figure 7.1 we selected 400×400 matrices A with random $\mathcal{N}(0,1)$ entries and modified their singular values. Then 20×5 matrices U and V with $\mathcal{N}(0,1)$ entries were sampled and from these 20×20 matrices were created by computing UV^T. The measurement vector $\mathbf{b}$ was created by computing $\mathbf{b} = AX + \epsilon$, where $\epsilon \in \mathcal{N}(0,\sigma^2)$ for varying noise level σ between 0 and 1.

In Figure 7.1 (a) we plotted the measurement fit $\|\mathcal{A}X - \mathbf{b}\|$ versus the noise level σ for the solutions obtained with (7.5) and (7.50). Note that (7.50) does not specify the sought rank, therefore we search iteratively for the smallest value of μ that gives the correct rank.

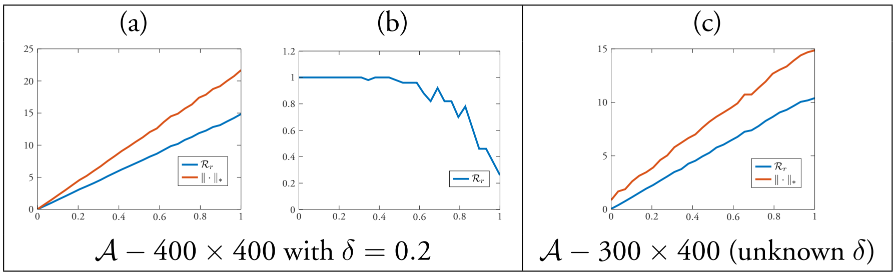


Figure 7.1: (a) - Noise level (x-axis) vs. data fit $\|\mathcal{A}X - \mathbf{b}\|$ (y-axis) for the solutions obtained with (7.5) and (7.50). (b) - Fraction of instances where the solution of (7.5) could be verified to be globally optimal. (c) - Same as (a). (a) and (b) uses 400×400 $\mathcal{A}$ with $\delta = 0.2$ while (c) uses 300×400 $\mathcal{A}$.

In Figure 7.1 we computed the Z matrix and plotted the fraction of problem instances where its singular values fulfilled $z_{r+1} < (1 - 2\delta)z_r$, with $\delta = 0.2$. For these instances the obtained stationary points are also globally optimal according to our main result.

Figure 7.1(c) is the same experiment as in (a), but instead an under determined matrix A of size 300×400. From [58] it is known that if A is $p \times mn$ and the elements of A are drawn from $\mathcal{N}(0, \frac{1}{p})$, then $\mathcal{A}$ fulfills (7.3) with high probability. The exact value of δ_q is however difficult to determine and therefore we are not able to verify optimality in this case.

7.3.4 Non-Rigid Structure from Motion

In this section we consider the problem of Non-Rigid Structure from Motion. We follow the approach of [25] and let

$$X = \begin{bmatrix} X_1 \\ Y_1 \\ Z_1 \\ \vdots \\ X_F \\ Y_F \\ Z_F \end{bmatrix} \text{ and } X^{\#} = \begin{bmatrix} X_1 & Y_1 & Z_1 \\ \vdots & \vdots & \vdots \\ X_F & Y_F & Z_F \end{bmatrix}, \tag{7.51}$$

where X_i, Y_i and Z_i are $1 \times m$ matrices containing the x-, y- and z-coordinates. of the tracked points in image i. Under the assumption of an orthographic camera the projection of the $3D$ points can be modeled using $M = RX$, where R is a $2F \times 3F$ block diagonal matrix with 2×3 blocks R_i, consisting of two orthogonal rows that encode the camera orientation in image i. The resulting $2F \times m$ measurement matrix M consists of the $x-$ and $y-$image coordinates of the tracked points. Under the assumption of a linear shape basis model [12] with r deformation modes, the matrix $X^{\#}$ can be factorized into $X^{\#} = CB$, where the $r \times 3m$ matrix B contain the basis elements. It is clear that such a factorization is possible when $X^{\#}$ is of rank r. We therefore search for the matrix $X^{\#}$ of rank r that minimizes the residual error $\|PX - M\|_F^2$. The linear operator defined by $\mathcal{A}(X^{\#}) = RX$ does by itself not obey (7.3) since there are typically low-rank matrices in its null space. This can be seen by noting that if N_i is the 3×1 vector perpendicular to the two rows of R_i then

$$\begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} = N_i C_i, \tag{7.52}$$

where C_i is any $1 \times m$ matrix, is in the null space of R_i. Therefore any matrix of the form

$$N^{\#}(C) = \begin{bmatrix} n_{11}C_1 & n_{21}C_1 & n_{31}C_1 \\ n_{12}C_2 & n_{22}C_2 & n_{32}C_2 \\ \vdots & \vdots & \vdots \\ n_{1F}C_F & n_{2F}C_F & n_{3F}C_F \end{bmatrix}, \tag{7.53}$$

where n_{ij} are the elements of N_i, vanishes under $\mathcal{A}$. Setting everything but the first row of $N^{\#}(C)$ to zero shows that there is a rank 1 matrix in the null space of $\mathcal{A}$. Moreover, if the rows of the optimal $X^{\#}$ spans such a matrix it will not be unique since we may add $N^{\#}(C)$ without affecting the projections or the rank.

In Figure 7.3 we compare the two relaxations

$$\mathcal{R}_r(X^{\#}) + \|RX - M\|_F^2 \tag{7.54}$$

and

$$\mu\|X^{\#}\|_* + \|RX - M\|_F^2. \tag{7.55}$$

on the four MOCAP sequences displayed in Figure 7.2, obtained from [25]. These consists of real motion capture data and therefore the ground truth solution is only approximately of low rank.

In Figure 7.3 we plot the rank of the obtained solution versus the data fit $\|RX - M\|_F^2$. Since (7.55) does not allow us to directly specify the rank of the sought matrix, we solved the problem for 50 values of μ between 1 and 100 (orange curve) and computed the resulting rank and data fit. Note that due to the shrinking effect, the solution of (7.55) is affected by a change of μ even if the rank is not changed. Consequently, to achieve data fit, we should as in the synthetic experiment choose the smallest μ that gives the desired rank.

Even though (7.3) does not hold, (7.54) consistently gives better data fit with lower rank than 7.55. Figure 7.4 shows the rank versus the distance to the ground truth solution. For high rank the distance (7.54) is typically higher than for (7.55). A feasible explanation is that when the rank is high it is more likely that the row space of $X^{\#}$ contains a matrix of the type $N(C)$. Loosely speaking, when we allow to much complex deformation it becomes more difficult to uniquely recover the shape. The nuclear norm's built in bias to small solutions helps to regularize the the problem when the rank constraint is not discriminative enough.

One way of handling the null space of $\mathcal{A}$ is to add additional regularizes that penalize low rank matrices of the type $N(C)$. Dai *et al.* [25] suggested to use the derivative prior $\|DX^{\#}\|_F^2$, where the matrix $D : \mathbb{R}^F \to \mathbb{R}^{F-1}$ is a first order difference operator. The null space of D contains matrices that are constant in each column. Since this implies that the scene is rigid it is clear that $N(C)$ is not in the null space of D. We add this term and compare

$$\mathcal{R}_r(X^{\#}) + \|RX - M\|_F^2 + \|DX^{\#}\|_F^2 \tag{7.56}$$

and

$$\mu\|X^{\#}\|_* + \|RX - M\|_F^2 + \|DX^{\#}\|_F^2. \tag{7.57}$$

Figures 7.5 and 7.6 show the result. In this case both the data fit and the distance to ground truth is consistently better with (7.56) than (7.57). When the rank increases most of the regularization comes from the derivative prior leading to both methods providing similar results.

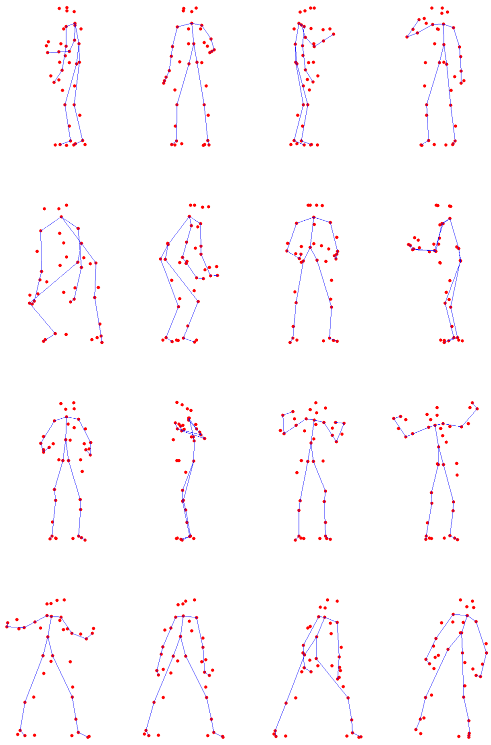


Figure 7.2: Four images from each of the MOCAP data sets.

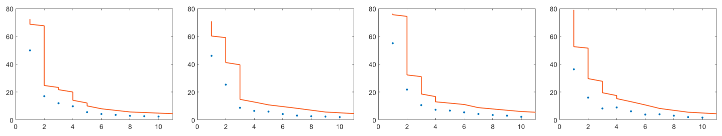


Figure 7.3: Results obtained with (7.54) (blue dots) and (7.55) (orange curve) for the four sequences. Data fit $\|RX - M\|_F$ (y-axis) versus rank$(X^{\#})$ (x-axis).

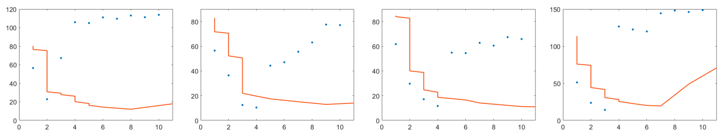


Figure 7.4: Results obtained with (7.54) (blue dots) and (7.55) (orange curve) for the four sequences. Distance to ground truth $\|X - X_{gt}\|_F$ (y-axis) versus rank$(X^{\#})$ (x-axis).

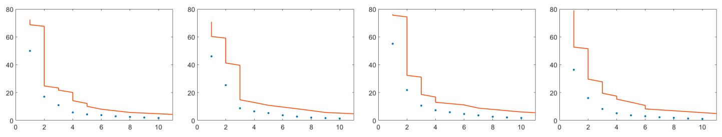


Figure 7.5: Results obtained with (7.56) (blue dots) and (7.57) (orange curve) for the four sequences. Data fit $\|RX - M\|_F$ (y-axis) versus rank$(X^{\#})$ (x-axis).

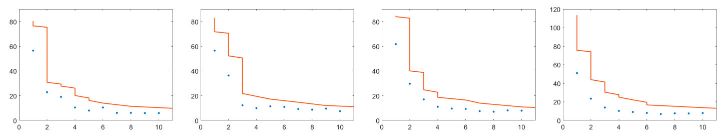


Figure 7.6: Results obtained with (7.56) (blue dots) and (7.57) (orange curve) for the four sequences. Distance to ground truth $\|X - X_{gt}\|_F$ (y-axis) versus rank$(X^{\#})$ (x-axis) is plotted for various regularization strengths.

7.4 Conclusion

In the last chapter we have studied the local minimum of a non-convex rank regularization approach. The main result is that if the RIP inequality (7.3) holds then there is often a unique local minimum. Our experimental evaluation shows that the proposed approach often gives better results than the nuclear norm relaxation, even when the RIP constraint does not hold.

Bibliography

[1] KinectFusion Implementation in the Point Cloud Library (PCL). URL `http://svn.pointclouds.org/pcl/trunk/`. 17, 36, 37, 38, 45

[2] *The MOSEK optimization toolbox for MATLAB manual.* URL `www.mosek.com`. 87

[3] F. Andersson, M. Carlsson, and C. Olsson. Convex envelopes for fixed rank approximation. *Optimization Letters*, 11(8):1783–1795, Dec 2017. 100, 101

[4] R. Angst, C. Zach, and M. Pollefeys. The generalized trace-norm and its application to structure-from-motion problems. In *International Conference on Computer Vision*, pages 2502–2509, Barcelona, Spain, 2011. IEEE. 59

[5] D. B. Kubacki, Q. H. Bui, S Derin B., and M. Do. Registration and integration of multiple depth images using signed distance function. In *The International Society for Optical Engineering*, pages 22–, Burlingame, USA, 2012. 18

[6] R. Basri, D. Jacobs, and I. Kemelmacher. Photometric stereo with general, unknown lighting. *International Journal of Computer Vision*, 72(3):239–257, May 2007. 58

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Computer vision and Image Understanding*, 110(3):346–359, june 2008. 50

[8] Dimitri P. Bertsekas. *Nonlinear programming.* Athena Scientific, 2nd edition, September 2008. 102

[9] P.J. Besl and N.D. McKay. A method for registration of 3-D shapes. *Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 12

[10] G. Blais and M.D. Levine. Registering multiview range data to create 3D computer objects. *Transactions on Pattern Analysis and Machine Intelligence*, 17:820–824, 1993. 12, 38

[11] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends in Machine Learning*, 3(1):1–122, 2011. 68, 90

[12] C. Bregler, A. Hertzmann, and H. Biermann. Recovering non-rigid 3D shape from image streams. In *Conference on Computer Vision and Pattern Recognition*, pages 690–696, Hilton Head, USA, 2000. 58, 110

[13] E. Bylow, J. Sturm, C. Kerl, F. Kahl, and D. Cremers. Real-time camera tracking and 3d reconstruction using signed distance functions. In *Robotic Science and System*, volume 9, Berlin, Germany, 2013. 2, 51

[14] E. Bylow, C. Olsson, and F. Kahl. Robust camera tracking by combining color and depth measurements. In *International Conference on Pattern Recognition*, pages 4038–4043, Stockholm, Sweden, 2014. 2, 53

[15] E. Bylow, C. Olsson, and F. Kahl. Robust online 3d reconstruction combining a depth sensor and sparse feature points. In *International Conference on Pattern Recognition, (ICPR)*, pages 3709–3714, Dec 2016. 2

[16] E. Bylow, C. Olsson, F. Kahl, and M. Nilsson. Minimizing the maximal rank. In *Conference on Computer Vision and Pattern Recognition*, Las Vegas, USA, 2016. 2

[17] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717 – 772, Apr 2009. 100

[18] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *Journal of the ACM*, 58(3):11:1–11:37, 2011. 59, 100

[19] D. Canelhas. Scene representation, registration and object detection in a truncated signed distance function representation of 3d space. Master's thesis, Örebro University, School of Science and Technology, 2012. 18

[20] M. Carlsson. On convexification/optimization of functionals including an l2-misfit term. *arXiv preprint arXiv:1609.09378*, 2016. 102

[21] Y. Chen and G. Medioni. Object modelling by registration of multiple range images. *Image and Vision Computing*, 10(3):145 – 155, 1992. 12, 13

[22] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Conference on Computer Graphics and Interactive Techniques*, pages 303–312, New York, USA, 1996. 12, 18, 21, 23

[23] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. BM3D image denoising with shape-adaptive principal component analysis. In *Workshop on Signal Processing with Adaptive Sparse Structured Representations*, Saint-Malo, France, 2009. 92, 93, 97

[24] A. Dai, M. Nießner, M. Zollhöfer, S. Izadi, and C. Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ACM Transactions on Graphics*, 36(3):24:1–24:18, 2017. 20

[25] Y. Dai, H. Li, and M. He. A simple prior-free method for non-rigid structure-from-motion factorization. *International Journal of Computer Vision*, 107(2):101–122, 2014. 99, 109, 111

[26] F. Endres, J. Hess, N. Engelhard, J. Sturm, D. Cremers, and W. Burgard. An evaluation of the rgb-d slam system. In *International Conference on Robotics and Automation*, pages 1691–1696, St Paul, USA, 2012. 19, 36, 37

[27] A. Eriksson and A. Hengel. Efficient computation of robust weighted low-rank matrix approximations using the L_1 norm. *Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1681–1690, 2012. 58

[28] Maryam Fazel, Haitham Hindi, and Stephen P Boyd. A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001.*, pages 4734–4739, 2001. 59

[29] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, 2003. 8

[30] R. Garg, A. Roussos, and L. Agapito. A variational approach to video registration with subspace constraints. *International Journal of Computer Vision*, 104(3):286–314, Sep 2013. 58

[31] P. Gong, C. Zhang, Z. Lu, J. Z. Huang, and J. Ye. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. In *International Conference on Machine Learning*, pages II–37–II–45, 2013. 108

[32] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. Second Edition. 3, 8

[33] M. Hein and M. Maier. Manifold denoising. In *Advances in Neural Information Processing Systems*, pages 561–568, Cambridge, USA, 2007. 81, 91, 92

[34] Berthold K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America*, 4(4):629–642, 1987. 12, 13

[35] Yao Hu, Debing Zhang, Jieping Ye, Xuelong Li, and Xiaofei He. Fast and accurate matrix completion via truncated nuclear norm regularization. *Transacions on Pattern Analysis and Machine Intelligence*, 35(9):2117–2130, 2013. 73

[36] J. J. Hull. A database for handwritten text recognition research. *Transactions on Pattern Analysis and Machine Intelligence*, 16(5):550–554, 1994. 91

[37] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: real-time 3d reconstruction and interaction using a moving depth camera. In *ACM symposium on User interface software and technology*, pages 559–568, New York, USA, 2011. 10, 17

[38] V. Jojic, S. Saria, and D. Koller. Convex envelopes of complexity controlling penalties: the case against premature envelopment. In *International Conference on Artificial Intelligence and Statistics*, pages 399–406, Fort Lauderdale, USA, 2011. 60

[39] C. Julià, F. Lumbreras, and A. Sappa. A factorization-based approach to photometric stereo. *International Journal of Imaging Systems and Technology*, 21:115 – 119, 03 2011. 58

[40] C. Kerl, J. Sturm, and D. Cremers. Robust odometry estimation for rgb-d cameras. In *International Conference on Robotics and Automation*, pages 3748–3754, Karlsruhe, Germany, 2013. 19

[41] C. Kerl, J. Sturm, and D. Cremers. Dense visual slam for rgb-d cameras. In *International Conference on Intelligent Robots and Systems*, pages 2100–2106, Tokyo, Japan, 2013. 19

[42] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010. 73

[43] R. Kuemmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. g2o: A general framework for graph optimization. In *International Conference on Robotics and Automation*, pages 3607–3613, Shanghai, China, 2011. 19, 45

[44] V. Larsson and C. Olsson. Convex low rank approximation. *International Journal of Computer Vision*, 120(2):194–214, Nov 2016. 100, 102, 103, 108

[45] V. Larsson, C. Olsson, E. Bylow, and F. Kahl. Rank minimization with structured data patterns. In *European Conference on Computer Vision*, pages 250–265. Zürich, Switzerland, 2014. 2, 94

[46] A. S. Lewis. The convex analysis of unitarily invariant matrix functions. *Journal of Convex Analysis*, 2(1/2):173–183, 1995. 70

[47] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, San Francisco, CA, USA, 1981. 79

[48] Y. Ma, S. Soatto, J. Kosecka, and S. Sastry. *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003. 27

[49] K. Mohan and M. Fazel. Iterative reweighted least squares for matrix rank minimization. In *Allerton Conference on Communication, Control, and Computing*, pages 653–661, Monticello, USA, 2010. 100

[50] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohli, J. Shotton, S. Hodges, and A.W. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, Washington DC, USA, 2011. 10, 17, 18, 19, 30, 36, 51

[51] C. Olsson and M. Oskarsson. A convex approach to low rank matrix approximation with missing data. In *Scandinavian Conference on Image Analysis*, pages 301–309, Oslo, Norway, 2009. 59

[52] C. Olsson, M. Carlsson, F. Andersson, and V. Larsson. Non-convex rank/sparsity regularization and local minima. In *International Conference on Computer Vision*, 2017. 101, 102, 104, 106

[53] Carl Olsson, Marcus Carlsson, and Erik Bylow. A non-convex relaxation for fixed-rank approximation. In *International Conference on Computer Vision Workshop*, pages 1809–1817, Venice, Italy, 2017. 2

[54] S. Osher and R. P. Fedkiw. *Level set methods and dynamic implicit surfaces*. Applied mathematical science. Springer, New York, USA, 2003. 10, 20

[55] S. Oymak, K. Mohan, M. Fazel, and B. Hassibi. A simplified approach to recovery conditions for low rank matrices. In *International Symposium on Information Theory Proceedings*, pages 2318–2322, St. Petersburg, Russia, 2011. 100

[56] S. Oymak, A. Jalali, M. Fazel, Y. C. Eldar, and B. Hassibi. Simultaneously structured models with application to sparse and low-rank matrices. *IEEE Transactions on Information Theory*, 61(5):2886–2908, 2015. 100

[57] H. Pfister, M. Zwicker, J. van Baar, and M. Gross. Surfels: Surface elements as rendering primitives. In *Conference on Computer Graphics and Interactive Techniques*, pages 335–342, New York, USA, 2000. 19

[58] B. Recht, M. Fazel, and P.A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52:471–501, August 2010. 59, 99, 100, 109

[59] C. Y. Ren and I. Reid. A unified energy minimization framework for model fitting in depth. In *European Conference on Computer Vision*, pages 72–82, Florence, Italy, 2012. 18

[60] R.T. Rockafellar. *Convex analysis*. Princeton Mathematical Series. Princeton University Press, Princeton, N. J., 1970. 85

[61] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *International Conference on 3D Digital Imaging and Modeling*, pages 145–152, Quebec City, Canada, 2001. 12

[62] S. Sengupta, H. Zhou, W. Forkel, R. Basri, T. Goldstein, and D. Jacobs. Solving uncalibrated photometric stereo using fewer images by jointly optimizing low-rank matrix completion and integrability. *Journal of Mathematical Imaging and Vision*, 2017. 58

[63] F. Steinbruecker, J. Sturm, and D. Cremers. Real-time visual odometry from dense rgb-d images. In *Workshop on Live Dense Reconstruction with Moving Cameras at International Conference on Computer Vision*, Barcelona, Spain, 2011. 14, 18, 19

[64] F. Steinbruecker, C. Kerl, J. Sturm, and D. Cremers. Large-scale multi-resolution surface reconstruction from rgb-d sequences. In *International Conference on Computer Vision*, Sydney, Australia, 2013. 10, 14, 45, 46, 51

[65] D. Strelow. General and nested Wiberg minimization. In *Conference on Computer Vision and Pattern Recognition*, pages 1584–1591, Providence, USA, 2012. 58

[66] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. A benchmark for the evaluation of RGB-D SLAM systems. In *International Conference on Intelligent Robots and Systems*, pages 573–580, Algarve, Portugal, 2012. 17, 31, 45, 51, 53

[67] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. Copyme3d: Scanning and printing persons in 3d. In *German Conference on Pattern Recognition*, pages 405–414. Saarbrücken, Germany, 2013. 2

[68] J. Sturm, E. Bylow, F. Kahl, and D. Cremers. Dense tracking and mapping with a quadrocopter. In *Unmanned Aerial Vehicle in Geomatics*, pages 371–376, Rostock, Germany, 2013. 2

[69] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11-12:625–653, 1999. 87

[70] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: A factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992. 55, 58

[71] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald. Robust real-time visual odometry for dense RGB-D mapping. In *International Conference on Robotics and Automation*, pages 5724–5731, Karlsruhe, Germany, 2013. 10, 19

[72] T. Whelan, S. Leutenegger, R. F Salas-Moreno, B. Glocker, and A. J. Davison. Elasticfusion: Dense slam without a pose graph. In *Robotics: Science and Systems*, page 1697–1716, Rome, Italy, 2015. 19

[73] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. *Autonomous Robots*, 34(3):189–206, 2013. 9, 19

[74] J. Yan and M. Pollefeys. A factorization-based approach for articulated non-rigid shape, motion and kinematic chain recovery from video. *Transactions on Pattern Analysis and Machine Intelligence*, 30(5):865–877, 2008. 58

[75] Y. Zheng, G. Liu, S. Sugimoto, S. Yan, and M. Okutomi. Practical low-rank matrix approximation under robust L_1-norm. In *Conference on Computer Vision and Pattern Recognition*, pages 1410–1417, Providence, USA, 2012. 58

Printed by Media-Tryck, Lund 2018 NORDIC SWAN ECOLABEL 3041 0903

LUND UNIVERSITY

Doctoral Theses in Mathematical Sciences 2018:2
ISBN 978-91-7753-622-2 (print)
LUTFMA-1066-2018
ISSN 1404-0034