Technische Universität München

Department of Mathematics

Master's Thesis

Joint Optimization of Pose And Depth Using a Prox-Linear Approach

Florian Hofherr

Supervisor: Prof. Dr. Daniel Cremers

Advisor: Nikolaus Demmel and Emanuel Laude

Submission Date: 15.12.2019

I assure the single handed composition of this masters's thesis only supported by declared resources.

Garching, December 13, 2019,

Florian Hofherr

Abstract

The problem of estimating a 3D scene and camera positions from images taken by this camera is known as simultaneous localization and mapping (SLAM) in robotics. Many state-of-the-art methods are based on a two-step approach where the pose and the depth are estimated sequentially. In this thesis, we examine a SLAM approach that jointly estimates the camera pose and a dense depth map. The approach uses a photometric data term and a regularization of the depth map. The resulting optimization problem is non-linear, non-convex and high-dimensional due to the dense approach. We solve this optimization problem using the prox-linear approach, which yields an iteration of sub-problems by repeated linearization of parts of the cost function. For the solution of the sub-problems, the primal-dual hybrid gradient (PDHG) approach is employed. We evaluate the performance of the method as well as the effect of different data loss functions using two synthetic data sets that contain a ground truth. The experiments show that the joint approach works well and that the results are comparable to the results of a pure depth or pose estimation.

Zusammenfassung

Die Rekonstruktion einer 3D Szene und von Kamerapositionen mittels Bilder, die von selbiger Kamera aufgenommen wurden, ist in der Robotik als Simultaneous Localization And Mapping (SLAM) bekannt. Viele moderne Methoden basieren auf einem zweistufigen Ansatz, in dem die Kamerapose und die Tiefenkarte sequenziell geschätzt werden. In der vorliegenden Arbeit wird ein SLAM Ansatz untersucht, in dem die Pose und eine dichte Tiefenkarte simultan geschätzt werden. Dabei wird ein photometrischer Datenterm verwendet und die Tiefenkarte wird zusätzlich regularisiert. Es resultiert ein nicht-lineares und nicht-konvexes Optimierungsproblem, welches durch den dichten Ansatz zudem sehr hochdimensional ist. Für die Optimierung wird der Prox-Linear Ansatz verwendet, welcher durch wiederholte Linearisierung von Teilen der Kostenfunktion eine Folge von Sub-Problemen liefert. Zur Lösung der Sub-Probleme findet der Primal-Dual Hybrid Gradient (PDHG) Ansatz Anwendung. Das Verhalten des Algorithmus sowie ein Vergleich von verschiedenen Daten Loss Funktionen werden anhand zweier synthetischer Datensets mit bekannter Ground Truth untersucht. Die Experimente zeigen, dass der simultane Ansatz gut funktioniert und die Ergebnisse vergleichbar zu Ansätzen zur reinen Posen- bzw. Tiefenschätzung sind.

Acknowledgments

First I, would like to thank my supervisors Nikolaus Demmel and Emanuel Laude for providing me with this exciting topic and for supervising my thesis. I highly appreciate the knowledge and experience they have shared with me and I am very thankful for the productive and enlightening talks and discussions.

Second, I would like to thank the supervisor of my bachelor thesis, Dr. Daniel Karrasch, from whom I have learned a lot about scientific writing and the value of neat figures - both of which is hopefully showing in this thesis. Moreover, he has become a mentor to me during my master studies, and I am very grateful for his support and advice.

Finally, I am very grateful to my family and my beloved fiancée Rebecca for their love and support. They have endured the hardships of my studies with me, and without their patience, understanding and encouragement I would have struggled a lot more.

Contents

1 Introduction

The ability of a technical system to understand its surroundings based on images taken by an onboard camera is crucial for an abundance of modern technologies. Prominent applications include autonomous driving of vehicles and robots, autonomous flying of aerial vehicles like drones, and reconstruction from aerial images. While a single image is never able to capture all the information of a scene since we are projecting from 3 dimensions to only two, a sequence of images of the same scene allows us to reconstruct a 3D model of that scene - often to an amazingly high accuracy. The approach of using subsequent images to reconstruct the scene is known as structure from motion in the computer vision [20]. In the robotics community, the term monocular SLAM (Simultaneous Localization and Mapping) is more common; see e.g. [10]. In this thesis, we use the latter terminology.

The general setting for SLAM is shown in Figure 1. We have a camera that moves around a scene and takes images from different positions and angles. In a general setting, we do not have any knowledge of the movement of the camera. The only data available are the images of the scene taken from unknown vantage points. The goal is to reconstruct both the scene as well as the positions and the rotations of the cameras. To do this, we need to model the process that created the images. By inverting this model, we hope to be able to recover the scene and the camera poses.

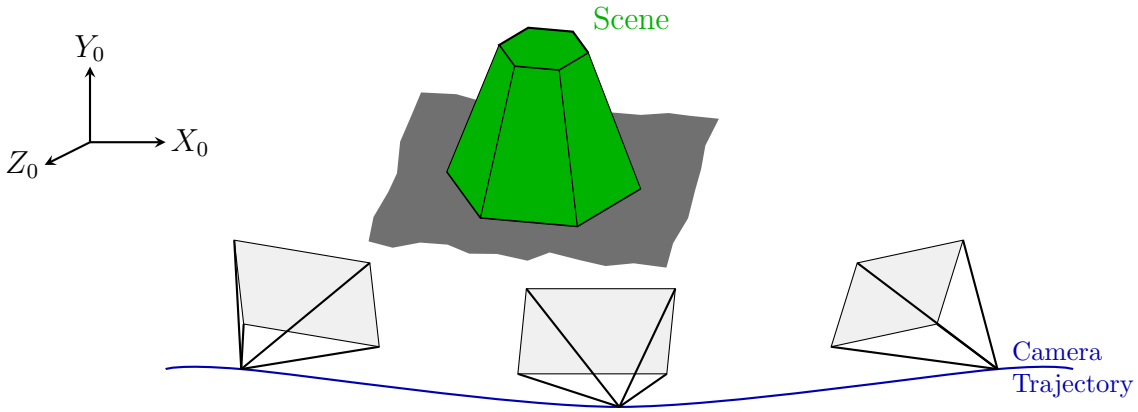


Figure 1: *Schematic of the SLAM problem. A camera moves through a scene and takes images from different (unknown) positions. The goal is to estimate a set of points describing the scene as well as the camera poses using the images as input data. A world coordinate system $X_0Y_0Z_0$ is used to describe positions and orientations.*

We can use an arbitrary world coordinate system $X_0Y_0Z_0$ to represent the camera poses (position and orientation). The scene is usually represented by a *depth map* defined in one of the camera frames, which we consider the *reference frame*. In the case of a digital image, the depth map assigns each pixel a depth value. This value is the distance from the camera to the structure in the real world that can be seen in this pixel. Note that by coordinate transformation, we could transform the depth map into a mesh of 3D world points in the reference frame. However, for applications like autonomous driving, the depth information in the camera coordinate system is much more useful, e.g. for obstacle avoidance.

While in the past years plenty of methods have been proposed, they can be categorized according to several criteria.

Geometric Methods Geometric models are based on a geometric error, i.e. we try to find 3D wold point locations that have the best consistency given the data. Working with geometric quantities requires a pre-processing step where we need to establish point correspondences between the images. This often involves a keypoint selection to obtain distinctive points that are easily traceable. Then the point positions and camera pose(s) can be estimated based on a geometric triangulation. The positions of the cameras correspond to additional constraints that confine the positions of the 3D points to lines. Therefore such methods are often referred to as *Bundle Adjustment*. Successful examples of this approach include PTAM [24], monoSLAM [11], ORB-SLAM [31].

The methods cited above are all sparse, i.e. we only estimate depth values for a subset of all pixels. Sparsity is partly required by the feature point based approach but also desired because it reduces the number of unknowns and therefore makes these methods strikingly fast. Still, there are also dense photometric methods in which a depth map for each pixel is estimated. Often regularized dense flow fields are used for this; see e.g. [44, 39].

Photometric Methods In recent years photometric models have gained popularity. In contrast to geometric methods, they work directly on the brightness of the images and minimize a photometric error, i.e. a difference in these brightness values. Therefore no pre-processing step is required as we try to align the images such that the intensity values are as consistent as possible. While geometric methods are considered *indirect* due to the pre-processing step, photometric methods are *direct* approaches as we are using the images directly. Again we can distinguish sparse and dense methods.

Dense photometric methods are prominent in smaller environments as most of them do not account for drifting effects that occur for longer camera movements. Successful approaches include [42, 32, 25] which all use a regularization term on the depth map.

To account for accumulating scale drifts, Engel et al. proposed the LSD-SLAM method [17], which uses loop closuring. However, to stay real-time capable, this method is only semi-dense, i.e. a depth map is only computed for pixels that show a sufficient change in the brightness value.

The DSO approach by Engel et al. further reduces the number of depth values that are considered and therefore is a sparse photometric approach [16]. It selects an evenly spread set of pixels for which depth values are to be estimated. To increase the robustness of the photometric error term, the approach uses a neighborhood pattern that evaluates the intensity difference for a system of neighboring pixels for each pixel that is to be assigned a depth value.

Methods Using Additional Data As many technical systems are equipped with a variety of sensors, often more than only the image data is available for the SLAM estimation problem. Inertial measurement units (IMUs) yield data on the movement of the systems, and RGB-D cameras or laser systems provide additional depth information to name just a few typical sensor technologies. This extra data can be

fused into the estimation process to speed up convergence and improve reliability. Successful approaches include [25], where a general method is proposed to incorporate sparse depth priors into a dense direct SLAM framework, and the work of Quan et al. who integrated IMU measurements into a SLAM system based on an extended Kalman filter [38].

Joint vs. Subsequent Estimation of Pose and Scene The estimated parameters can be split up into the pose and the depth. Many direct and dense approaches estimate those two parameter sets subsequently [42, 25]. In the first stage, they use a camera tracking method to estimate the camera movement. Those pose estimation approaches are often sparse and feature-based and include a check for loop closures. For example Stühmer [42] used PTAM [24] while Kuschk [25] employed LSD-SLAM [17]. Based on the estimated pose, the depth map is estimated in a subsequent step. In contrast to those two-stage methods, there are approaches like DSO which optimize pose and depth map jointly [16].

Contributions and Outline

In this thesis, we examine a regularized dense photometric SLAM approach that jointly optimizes pose and depth map. The method extends the work by Stühmer and Kuschk by including the pose in the estimation. For the resulting non-linear and non-convex optimization problem, the prox-linear approach, also called prox-descend and the primal-dual hybrid gradient (PDHG) method are used. The focus of this work is on proving the concept rather than on optimizing performance.

Section 2 covers the modeling of the image formation process. After the definition of the term "image" we proceed by the derivation of the pinhole camera model. Next, we address the topic of camera poses before concluding the chapter by introducing the warping function that relates multiple images of the same scene with each other.

In Section 3, we derive the cost function used for the estimation. First, we discuss different aspects related to the photometric data term, which includes a section on different loss functions. We then proceed by examining several choices for the regularization term before stating the SLAM optimization problem.

Section 4 is devoted to the optimization methods used to solve the SLAM problem. For the prox-linear and the PDHG algorithm, all relevant concepts and results are stated with references for the proofs.

Section 5 addresses the application of the optimization algorithms to the SLAM problem as well as the aspects relevant for the implementation. The result of this chapter is a pseudo-code, including all steps involved in solving the SLAM problem. To be able to evaluate the performance of the main approach, we are discussing a special case that can be solved partly analytically.

In Section 6, the numerical results are presented. First, we evaluate the performance of the joint approach using two different data sets. Second, we compare different data loss terms for the estimation. Finally, we investigate how scale drift effects affect the approach.

2 Modeling the image formation process

As we want to reverse the image formation process to recover the 3D scene and the camera poses, we need to have a model of this process. In this chapter, we are discussing the relevant concepts. After fixing the notion of an *image* for this thesis, we start by modeling the camera. After making some assumptions, we derive the model of the *pinhole camera*. Then we discuss how the same scene is observed in two different cameras. Finally, by combining the previous results, we obtain the *warping function* that describes the mapping from a point in the image plane of one camera to the image plane of a second camera, given a depth map and the relative positions of the camera. This warping function will be an integral part of the estimation algorithm. Unless stated otherwise, most of this chapter is based on the book by Ma et al. [28] with extensions and modifications of notations where it was deemed necessary for this thesis.

2.1 On Images and Depth Maps

A camera captures images by directing light onto an imaging surface where the incoming radiant flux (*irradiance*) is used to form the image. Accumulating the irradiance over the *exposure time* results in the *image intensity* or *brightness*. As the brightness corresponds to the incoming energy, it is a non-negative quantity. Mathematically, an image can be modeled as a function that assigns a brightness to each point of the image domain Ω.

In this thesis, we are dealing with images obtained from digital cameras exclusively. In a digital camera, the imaging surface contains an image sensor that divides the surface into a grid of many small sensors. Each small sensor integrates the irradiance reaching its surface over the exposure time to obtain a single brightness value for the whole sub-surface. The small sub-surfaces are referred to as *pixels*, and the number of pixels of the image sensor defines the resolution of the image - the more pixel we have, the finer the discretization is. We assume the pixels to be square.

The information stored as the *brightness value* depends on the type of the image. In the case of a grayscale image, the brightness is just a scalar value. For an RGB image, 3 values are stored to describe the irradiance in the red, the green, and the blue band of the spectrum. This concept can be extended further if the sensor of the camera is capable of capturing other parts of the spectrum as well. In general, the sensor captures c values per pixel, and we refer to that number as *channels*.

For storage reasons, the brightness values in digital images are usually quantized as well. A common interval is $[0, 255] \subset \mathbb{N}_0$ as this allows to store each brightness value in an unsigned 8-bit variable. However, as we are doing calculations with these values, we treat them as positive real numbers in the context of this thesis.

To sum it up, a digital image stores brightness vectors with c entries for each pixel. Therefore, we can represent a digital image taken by an image sensor with $m \times n$ pixels as a matrix $I \in \mathbb{R}_+^{m \times n \times c}$, where each value corresponds to the integrated irradiance measured by one pixel in the respective channel.

We now equip the image domain with a right-hand coordinate system. With a suitable scaling, we can establish a direct connection to the matrix representation of a digital image. Consider Figure 2, which shows part of the pixel grid on the image domain Ω and three common choices of coordinate systems. Note that for all

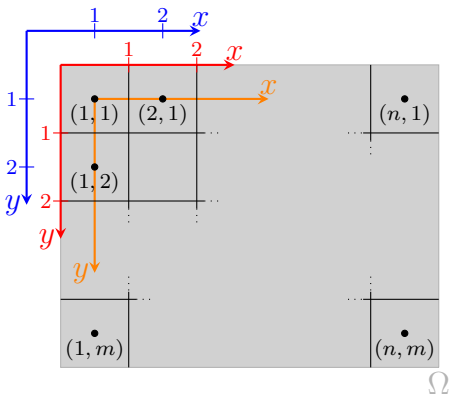


Figure 2: *Pixel grid on an image domain Ω and three possible choices of image coordinate systems. The red system has its origin on the top left corner of the image while the origin of the blue system is half a pixel north-west of this corner. The origin of the orange coordinate system is in the center of the top-left pixel. The units in all images are 1 pixel. We index the pixels by tuples of column and row to be consistent with the axes.*

choices, the x axis aligns with increasing column number, and the y axis aligns with increasing row number. As we usually write the x coordinate first, we index a pixel by the tuple (column, row), as indicated in the figure. Note that this leads to a somewhat unfamiliar indexing if we represent the image by a matrix as for matrices, usually the row is given first.

First, consider the blue coordinate system. The origin is half a pixel north-west of the top left corner of the image, and we have chosen the units to be 1 pixel. The result is that the pixel centers coincide with the integer points in the coordinate system, which makes addressing pixels very intuitive in this system. The drawback, however, is that care must be taken at the boundaries. At the left and the top side of the image, points having at least one component between 0 and 0.5 are not in the image domain despite being positive. Likewise, at the right and the bottom side points with coordinates up to 0.5 bigger than the number of pixels in the respective direction are still in the image domain despite being bigger than the number of pixels.

To avoid the problems at the boundary, some authors prefer the red coordinate system shown in Figure 2. The origin is at the top left corner, and the units are 1 pixel again. Now the image domain is bounded by 0 and the number of pixels in the respective direction, which makes the boundaries more intuitive. However, the pixel centers are not aligned with the integer points anymore but shifted half a pixel north-west of the respective point, which is the downside of this choice.

The orange coordinate system represents a third choice where the origin is placed in the pixel center of the top-left pixel. This choice has advantages in programming languages that use 0-indexing, like C++ since the integer points in the coordinate system coincide with the indices of the matrix representation of the image.

As we will need to check if points are within the image domain, we use the second (red) coordinate system in this thesis, thereby also staying consistent with [28]. Note that we still use the indexing of the pixels as shown in Figure 2, e.g. for addressing elements in the image matrix.

Having chosen the coordinate system, we now define the mapping that assigns an arbitrary point in the image plane the respective intensity value.

Definition 2.1 (Intensity Mapping in a Digital Image)**.** Let $I \in \mathbb{R}_+^{m\times n\times c}$ be a digital image with c channels and $\mathbf{x} = (x, y) \in \Omega$ a point in the respective image domain equipped with the red coordinate system in Figure 2. Then the brightness value for this point is given by

$$I(\mathbf{x}) = \left[I_{\lceil y\rceil\lceil x\rceil 1}, \ldots, I_{\lceil y\rceil\lceil x\rceil c}\right]^T$$

Where $\lceil x\rceil$ is the ceiling function that returns the smallest integer number that is bigger or equal to x and I_{ijk} is the entry in the i-th row, the j-th column on the k-th page of I.

We need the set of coordinates of all pixel centers which for the red coordinate system reads

Definition 2.2. We denote the set of all pixel centers as

$$\mathcal{P} = \{(x, y) \mid x \in \{0.5, 1.5, \ldots, n - 0.5\}, y \in \{0.5, 1.5, \ldots, m - 0.5\}\}$$

The Depth Map

Analogously to the intensity, we can assign other quantities to the pixels. In this thesis, we are interested in *depth maps*, i.e. we want to assign each pixel the z-component of the 3D world point that is visible in this pixel given in the camera coordinate system. Note that as for the brightness, this involves an averaging operation since we do not see only one single point in a pixel. As the distances are positive values, the depth map can be interpreted as an image with one channel. In this thesis, we refer to a depth map as matrix $\mathbf{h} \in \mathbb{R}_+^{m\times n}$, where we use an analogous mapping to the intensity mapping in Definition 2.1 to assign a depth value to an arbitrary point $x \in \Omega$.

2.2 The Camera Model

A standard camera is equipped with one or more lenses to focus the light onto the imaging surface. Modern camera systems often have a complex system of multiple lenses that serve a variety of purposes. Field of view and light sensitivity are only two of many factors that are influencing the design of the optical system of a camera system. Chapter 6.2 in the book by Born et al. [3] gives an overview of some classical lens systems that are used for cameras.

Building a model for such lens systems is possible but highly complex. See the first chapters and in particular, chapter 4 in [3] for a thorough discussion of the underlying optical theory. As we need to invert the camera model as well as evaluate it repeatedly, we need to make some assumptions to obtain a simpler model that is computationally feasible.

Even the *thin lens model*, which is already a substantial simplification, does not quite suit our purposes. For a discussion of this model and in particular, a distinction to modeling thick lenses; see e.g. [19].

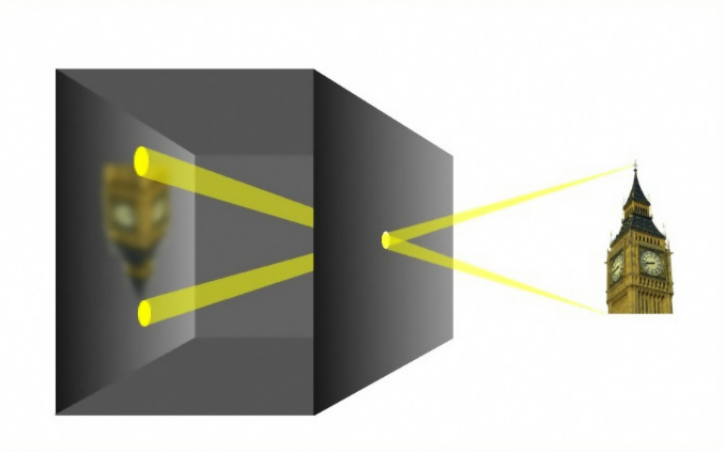


Figure 3: *Working principle of the camera obscura. The small hole (pinhole) in the right wall of the box focuses the light rays originating from Big Ben onto the left wall of the box, yielding an upside-down image. As the pinhole has a finite diameter, the rays originating from a single point on Big Ben reach multiple points on the image plane, which results in a blurring of the image.*
Source image: `https://commons.wikimedia.org/wiki/File:Lochkamera_prinzip.jpg`, *September 2019*

2.2.1 The Pinhole Camera

The pinhole camera model is motivated by the *camera obscura*, one of the first camera designs. A small hole (pinhole) in one of the sides of a box is used to focus the light rays emitted from an object onto the opposite wall of the box, which is the imaging surface, see Figure 3. Note that the image appears upside down. Since the pinhole has a finite diameter, the image is blurry as the rays emitted from one point on the object reach multiple points on the imaging surface. However, if we could reduce the pinhole to a single point, only one ray per point on the object would reach the imaging surface, which would result in a sharp image. Although this process is physically impossible to realize, it still yields a convenient model that is a sufficient approximation of a well-focused imaging system [28, section, 3.2.2] and therefore is one of the standard models in computer vision [16], [32], [42].

We start modeling the pinhole camera by defining a local camera coordinate system XYZ in the pinhole point which we will call *optical center* o from now on, see Figure 4. We define the *optical axis* as the line through o that is perpendicular to the image plane and orient the Z-axis along the optical axis. The X and the Y-axis are oriented parallel to the width and the height of the image plane. As we see shortly, this choice is the link to the image coordinate system already mentioned in Section 2.1. The distance between the optical center o and the image plane is referred to as the focal length f of the pinhole camera.

We define a (preliminary) image coordinate system $\xi\eta$ by projecting the X and the Y axis to the image plane. By using similar triangles, it is easy to see that a 3D world point $\mathbf{X}$ in the local camera coordinate system XYZ is related to its 2D image point $\mathbf{x}_{\xi\eta} = [\xi, \eta]^T$ in the image coordinate system via the so-called *perspective projection.*

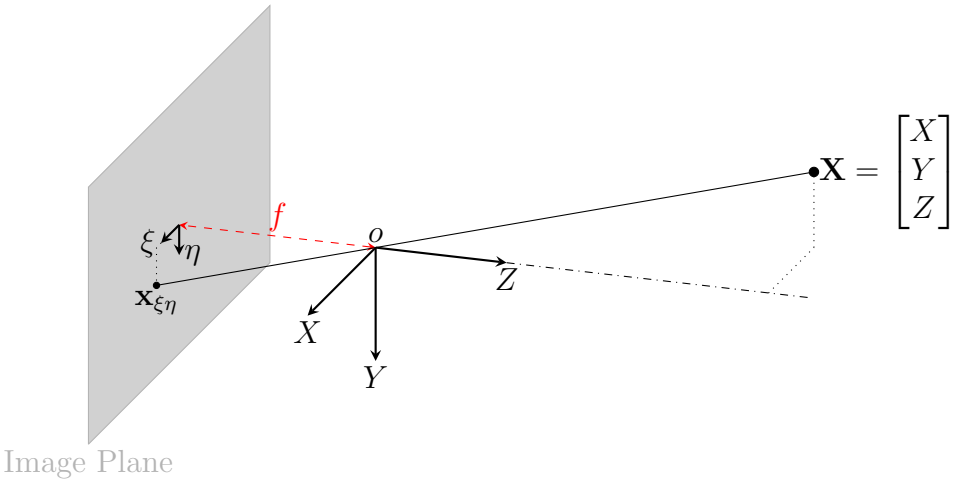


Figure 4: *The pinhole camera model. The local camera coordinate system XYZ is located in the optical center o with the Z-axis along the dash-dotted optical axis perpendicular to the imaging plane. The X and the Y axis are parallel to the width and the height of the image plane, respectively. The distance of the optical center to the image plane is the focal length f. The image coordinate system $\xi\eta$ is the projection of the X and the Y-axis to the image plane. The image of the world point $\mathbf{X}$ is at the intersection of the ray from $\mathbf{X}$ through o with the image plane. If we observe the image plane in Z direction (from behind) we see an upside-down and mirror-inverted image of the world scene.*

$$\begin{aligned} \xi &= -f\frac{X}{Z} \\ \eta &= -f\frac{Y}{Z} \end{aligned} \tag{2.1}$$

If we observe the image plane of the current model in Z direction (from behind), the real-world scene is mapped upside down and mirror-inverted onto the image plane, which is also showing in the negative signs in Equation (2.1). Images obtained from digital camera systems are usually already flipped back. In the formula, we can avoid these flips and therefore eliminate the minus signs by virtually placing the image plane in front of the optical center with a distance of f, as shown in Figure 5. It is easy to see that we obtain the same image as before but rotated by 180 degrees, so if we now observe the image plan in Z direction, the image is oriented correctly. The following formulas are referred to as the *frontal pinhole model* and are used in the thesis.

$$\begin{aligned} \xi &= f\frac{X}{Z} \\ \eta &= f\frac{Y}{Z} \end{aligned} \tag{2.2}$$

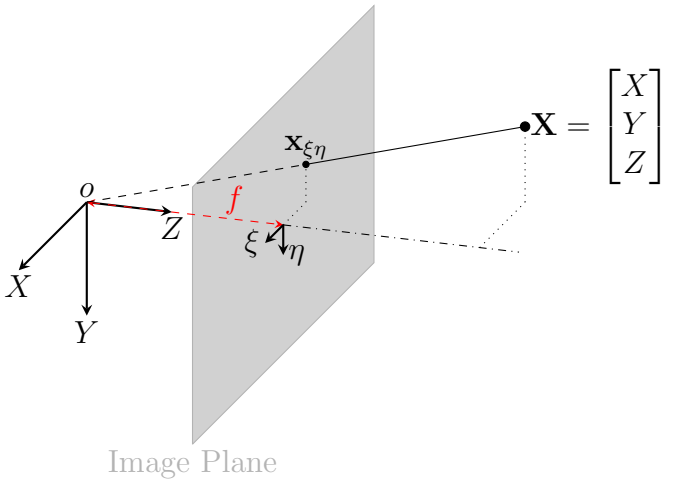


Figure 5: *The frontal pinhole camera model. The image plane is (virtually) placed in front of the opitcal center at a distance of f. If we observe the image plane in Z direction (from behind) we see a correctly oriented image of the world scene.*

Using homogeneous coordinates for the image point, we can rewrite Equation (2.2) in a matrix-vector form that we use frequently.

$$\underbrace{Z}_{:=h} \begin{bmatrix} \xi \\ \eta \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = K_f \mathbf{X} \tag{2.3}$$

As the Z coordinate is the depth of the 3D world point in the respective camera frame, i.e. the value of the depth map, which will be part of our optimization variable, we denote it by h from now on. Note that we re-name only the component that we multiplied to the left. The vector on the right that also contains Z is a representation of the actual 3D point that is replaced later. Therefore it remains unchanged.

In some cases, it can be helpful to represent the 3D point $\mathbf{X}$ in homogeneous coordinates as well. Therefore we introduce the *canonical projection matrix* Π_0.

$$h \begin{bmatrix} \xi \\ \eta \\ 1 \end{bmatrix} = K_f \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = K_f \Pi_0 \tilde{\mathbf{X}} \tag{2.4}$$

We denote a point in homogeneous coordinates by a tilde sign, i.e. for $\mathbf{v} \in \mathbb{R}^n$ we define $\tilde{\mathbf{v}} \in \mathbb{R}^{n+1}$ as

$$\tilde{\mathbf{v}} = \begin{bmatrix} \mathbf{v} \\ 1 \end{bmatrix}$$

2.2.2 Intrinsic Camera Parameters

So far the projected point is represented in the $\xi\eta$ coordinate system. We now want to obtain the representation in the coordinate system introduced in Section 2.1. See Figure 6 for a visualization of both coordinate systems. Note that since we are observing the image in Z direction, the axes are already oriented the correct way.

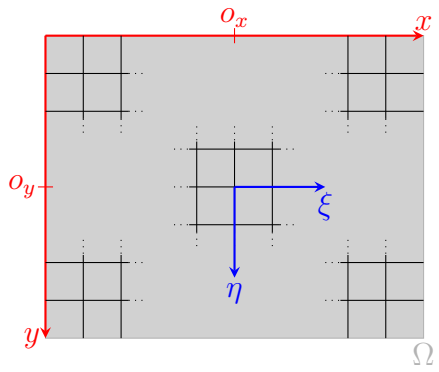


Figure 6: *The pixel image coordinate system xy in red and the world units coordinate system $\xi\eta$ in blue. The origin of the blue coordinate system in the xy system is denoted by (o_x, o_y). Note that this point is not necessarily at the common vertex of 4 pixels.*

The transformation between the coordinate systems is given by a shifting and a re-scaling from world units to pixel units and can be written as

$$\begin{aligned} x &= s_x \xi + o_x \\ y &= s_y \eta + o_y \end{aligned} \tag{2.5}$$

where s_x and s_y are the scaling parameters that transform world units to pixel units and o_x and o_y are the coordinates of the image coordinate system $\xi\eta$ in the image pixel coordinate system xy, see Figure 6.

Using homogeneous coordinates we can write the transformation in Equation (2.5) as the following matrix vector operation.

$$\tilde{\mathbf{x}} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \xi \\ \eta \\ 1 \end{bmatrix} = K_s \begin{bmatrix} \xi \\ \eta \\ 1 \end{bmatrix} \tag{2.6}$$

The so-called skew factor s_θ is a way to generalize the transformation and can be used if the pixel coordinate axes are not perpendicular to each other. In that case, the factor is proportional to $\cot(\theta)$, where θ is the angle between the axes. However, in this thesis, as well as in many relevant applications, the axes are perpendicular, which results in $s_\theta = 0$.

We now can combine Equation (2.3) and Equation (2.6) to obtain the projection from real world coordinates in the local camera coordinate system to image pixel coordinates

$$h\tilde{\mathbf{x}} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \underbrace{K_s K_f}_{:=K} \mathbf{X} \tag{2.7}$$

The matrix K is called *intrinsic parameter matrix* and can be obtained by calibration of the camera. The calibration process usually involves solving a parameter estimation problem where the variables are the intrinsic parameters and the input data are images of scenes with known patterns like checkerboards, see e.g. [19, section 1.3].

Figure 7: *On the left an image which shows radial distortion. On the right the same image agfter the undistortion.*
Source image: `https://github.com/bbenligiray/lens-distortion-rectification`, *October 2019*

2.2.3 Radial distortion

As stated before, the pinhole camera model is an idealization that standard cameras fulfill only to a certain extent. One major issue that we encounter frequently is radial distortion, an effect that is present in particular in images taken by cameras with a short focal length and cheap consumer electronic cameras. The effect causes straight lines to appear curved in the resulting image. For an example of the effect, see Figure 7. Note that in the example, straight lines appear curved towards the outside, which is known as *barrel distortion*. Curving towards the inside is also possible, which is known as *pincushion distortion*.

We can remove radial distortion from images in a pre-processing step. The results are undistorted images that follow the pinhole camera model much better. The distortion is caused by a radial change of the magnification around a *center of distortion*. We can model this by extending the pinhole model by a radial shift function that affects the projected points obtained by Equation (2.2). Depending on the application, we can write the shift as a function that takes the distorted coordinates and returns the undistorted ones or vice versa. For an undistortion of the image, the latter approach is better suited.

We can add a model for radial distortion to the projection from real world coordinates to pixel coordinates given in Equation (2.7) in the following way.

$$\tilde{x} = K \begin{bmatrix} L\left(\pi\left(\mathbf{X}\right)\right) \\ 1 \end{bmatrix} \tag{2.8}$$

where π denotes the generic projection $\pi\left(X\right) = \left[\frac{X}{Z}, \frac{X}{Z}\right]^T$ and L is the distortion function. A radial distortion centered at $c \in \mathbb{R}$ is given by the function

$$\begin{aligned} L\left(\mathbf{x}\right) &= c + f\left(\left\|\mathbf{x} - c\right\|\right)\left(\mathbf{x} - c\right) \\ f\left(r\right) &= 1 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 \end{aligned} \tag{2.9}$$

where f is one possible choice for a distortion factor with the distortion parameters a_i. To undistort an image we compute for each pixel in the undistorted image the location in the distorted image and assign the undistorted pixel the brightness at the location in the distorted image, which is usually interpolated bilinearly for that.[1]

[1] Bilinear interpolation will be discussed in Section 3.1.3

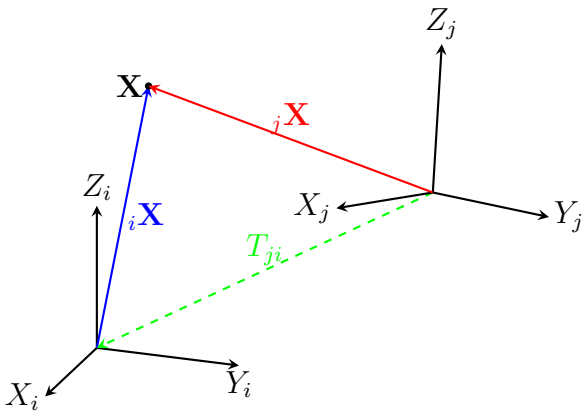


Figure 8: *Two coordinate systems i and j that haven an arbitrary orientation w.r.t each other. The general point $\mathbf{X}$ has the coordinate representations ${}_i\mathbf{X}$ and ${}_i\mathbf{X}$ in the respective coordinate system. The origin of the i-system given in the j-system is denoted by $\mathbf{T}_{ji}$.*

Given a pixel $\mathbf{x}$ in the undistorted image, the location in the distorted image $\mathbf{x}_{dist}$ can be calculated as

$$\mathbf{x}_{dist} = L\left(\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\end{bmatrix} K^{-1}\begin{bmatrix}\mathbf{x}\\ 1\end{bmatrix}\right) \tag{2.10}$$

The distortion parameters are obtained by calibration, see e.g. [12].

2.3 Camera Poses

To make use of the information contained in multiple images of the same scene, we need to understand how this scene is mapped to the different camera frames. To be able to use our pinhole camera model to project an arbitrary 3D point into different camera frames, we need to be able to represent this point in the different local camera coordinate systems. The change between the representation in different coordinate systems is achieved by a coordinate transformation.

2.3.1 Coordinate Transformations

The general setting is shown in Figure 8. We have an arbitrary 3D point $\mathbf{X}$ given in the coordinates of the system $X_iY_iZ_i$, and we want to represent this point in the coordinates of a system $X_jY_jZ_j$ that is rotated and translated with respect to the i system. We indicate the coordinate system that a point is represented in by a subscript to the left. The transformation of ${}_i\mathbf{X}$ to the representation in system j is given by a translation and rotation, and we can write it as

$${}_j\mathbf{X} = R_{ji}\,{}_i\mathbf{X} + T_{ji} \tag{2.11}$$

where $T_{ji} \in \mathbb{R}^3$ is a translation vector and $R_{ji} \in SO\,(3)$ is a rotation matrix. The space $SO\,(3)$ is the *special orthogonal group* of $\mathbb{R}^3$ which contains rotation matrices and is defined as

$$SO\,(3) = \left\{R \in \mathbb{R}^{3\times 3} \mid R^T R = I, \det\,(R) = +1\right\}$$

The matrix R_{ji} rotates the basis vectors of the coordinate system i to the system j. Therefore, the columns of R_{ji} consist of the basis vectors of the system i represented in the system j. The translation vector T_{ji} accounts for the difference between the origins of the two coordinate systems and is the position of the origin of the coordinate system i given in the coordinates of the system j.

In the SLAM context we will refer to the tuple (R_{ji}, T_{ji}) as pose of camera j relative to camera i or simply as *relative pose*. For readability, we omit the indices whenever it is clear in which coordinate systems we are working. For poses relative to the reference frame 0, we only give the index of the second coordinate system.

The transformation in Equation (2.11) is still an affine transformation. By using homogeneous coordinates, we can write it as a matrix-vector operation, which is useful in some cases.

$$\begin{bmatrix} {}_j\mathbf{X} \\ 1 \end{bmatrix} = \begin{bmatrix} R_{ji} & T_{ji} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} {}_i\mathbf{X} \\ 1 \end{bmatrix} \tag{2.12}$$

Remark. The tuple (R, T) that defines the transformation can also be identified as an element of the *special Euclidean group* $SE(3)$ which is defined as

$$SE(3) := \left\{ g = (R, T) \mid R \in SO(3), T \in \mathbb{R}^3 \right\}$$

Elements of this group are also known as *rigid body transformations*, which intuitively means that the object undergoing the transformation does not deform. The group structure ensures that the concatenation of two rigid body transformation is again a rigid body transformation and that we can invert any transformation.

2.3.2 Parametrization of the rotation matrix R

Although the matrix R has 9 entries, the conditions resulting from $R \in SO(3)$ leave only 3 of them as true free parameters. Therefore we need a parametrization of the matrix that ensures that these conditions hold. We are using *Rodrigues' Formula*.

A complete proof of this formula, as well as a rigorous definition of the concepts involved, is beyond the scope of this thesis, and we refer to [28, section 2.3.2]. The basic idea is to identify $SO(3)$ as a Lie group for which the corresponding Lie algebra is the space of 3×3 skew-symmetric matrices denoted by $so(3)$.[2]Rodrigues' formula is a closed representation of the exponential map

$$\exp : so(3) \to SO(3); \quad \widehat{\mathrm{w}} \mapsto e^{\widehat{\mathrm{w}}} \tag{2.13}$$

from the Lie algebra to the Lie group.

Theorem 2.1 (Rodrigues' Formula)**.** Let $R \in SO(3)$ be an arbitrary rotation matrix. Then there is a $\mathrm{w} \in \mathbb{R}^3$ such that

$$R = e^{\widehat{\mathrm{w}}} = I + \frac{\widehat{\mathrm{w}}}{\|\mathrm{w}\|} \sin(\|\mathrm{w}\|) + \frac{\widehat{\mathrm{w}}^2}{\|\mathrm{w}\|^2} (1 - \cos(\|\mathrm{w}\|))$$

[2]Simply speaking, a Lie group is a group that is also a smooth manifold. The Lie algebra of a Lie group is the tangential space at the identity element of the Lie group. Rigorous definitions for Lie groups, Lie algebras and the exponential map can be found in any textbook on differential geometry, e.g. [26].

where $\widehat{\cdot}$ is the hat operator which for $\mathrm{w} \in \mathbb{R}^3$ is defined as

$$\widehat{\mathrm{w}} = \begin{bmatrix} 0 & -\mathrm{w}_3 & \mathrm{w}_2 \\ \mathrm{w}_3 & 0 & -\mathrm{w}_1 \\ -\mathrm{w}_2 & \mathrm{w}_1 & 0 \end{bmatrix}$$

The vector w is not unique.

The inversion of the exponential map which is known as the *logarithm* is used to find the (non-unique) vector w given the rotation matrix R.

Theorem 2.2 (Logarithm of $SO(3)$)**.** Let $R \in SO(3)$ be an arbitrary rotation matrix. Then the (not necessarily unique) vector $\mathrm{w} \in \mathbb{R}^3$ such that $\exp(\hat{\mathrm{w}}) = R$ is given by

$$\mathrm{w} = \frac{\|\mathrm{w}\|}{2\sin(\|\mathrm{w}\|)} \begin{bmatrix} r_{32} - r_{23} \\ r_{13} - r_{31} \\ r_{21} - r_{12} \end{bmatrix} \quad \text{with} \quad \|\mathrm{w}\| = \cos^{-1}\left(\frac{\operatorname{trace}(R) - 1}{2}\right) \tag{2.14}$$

where r_{ij} denotes the entries of the matrix R. We write $\mathrm{w} = \log(R)$ to abbreviate the formulas above.

The vector w also has the following physical interpretation. The rotation described by R is the rotation around the axis w by the angle $\|\mathrm{w}\|$ in radians. This also provides an intuitive understanding of the non-uniqueness. We can always scale the w such that the length changes additively by an integer multiple of 2π. This does not change the rotation axis and simply adds a full rotation. The result, however, stays the same.

2.3.3 Linearization of Functions of Rotation Matrices

Many iterative optimization algorithms for non-linear problems depend on a linearization at the current iterate (think e.g. Gauss-Newton). Based on the linearization, we are then calculating a step on the optimization variable to obtain a new iterate. Due to the group properties of the special orthogonal group, there are two different possibilities to linearize a function of a rotation matrix.

Recall the linearization in the standard Euclidean setting. Consider a function $f : \mathbb{R}^m \to \mathbb{R}^n$ which we want to linearize at the point x^k. Using Taylor's theorem we can approximate the value of f at $y = x^k + \delta$ for a small increment $\delta \in \mathbb{R}^m$ by

$$f(y) = f\left(x^k + \delta\right) \approx f\left(x^k\right) + Jf\left(x^k\right)\delta \tag{2.15}$$

where Jf is the Jacobian matrix of f, see Appendix A for the definition used in this thesis.

We now transfer this concept to functions of rotation matrices. Consider the function $g : SO(3) \to \mathbb{R}^n$. We want to linearize this function a the point $R^k \in SO(3)$. Using Rodrigues' formula to parametrize the rotation matrix we can also consider the function $\tilde{g} : \mathbb{R}^3 \to \mathbb{R}^n$ with $\tilde{g}(\mathrm{w}) = g(\exp(\widehat{\mathrm{w}}))$. The linearization point of this function is $\mathrm{w}^k \in \mathbb{R}^3$ which is the corresponding parametrization of the rotation matrix, i.e. $R^k = \exp\left(\widehat{\mathrm{w}}^k\right)$.

Increment in the Lie Algebra In this case we are working with the function $\tilde{g}(\mathrm{w}) = g(\exp(\widehat{\mathrm{w}}))$ which means that the increment δ is done in a standard Euclidean space. Therefore this linearization is just the standard approach with a composition of functions. For $y = \exp\left(\widehat{w^k + \delta}\right)$ we can approximate

$$g(y) = \tilde{g}\left(\mathrm{w}^k + \delta\right) \approx \tilde{g}\left(\mathrm{w}^k\right) + J_{\mathrm{w}}\tilde{g}\left(\mathrm{w}^k\right)\delta \tag{2.16}$$

Note that for the Jacobian matrix we need to follow the chain rule, i.e.

$$J_{\mathrm{w}}\tilde{g}\left(\mathrm{w}^k\right) = J_R g\left(R^k\right) J_{\mathrm{w}} e^{\hat{\mathrm{w}}}\left(\mathrm{w}^k\right) \tag{2.17}$$

When we have obtained an increment δ by the algorithm the update of the variable is simply $\mathrm{w}^{k+1} = \mathrm{w}^k + \delta$.

Increment in the Lie Group Alternatively we can be work with the function g directly. In this case defining an increment is not straightforward, as we do not have $+$-operation in the group $SO(3)$. It is common to define the *encapsulation operator*.

$$\boxplus : SO(3) \times \mathbb{R}^3 \to SO(3); \quad R \boxplus \mathrm{w} = e^{\hat{\mathrm{w}}} R \tag{2.18}$$

This can be seen as a disturbance of the rotation matrix R. A detailed discussion of the encapsulation operator with the extension to general manifolds can be found e.g. in [21]. Fixing the linearization point we now consider the function $f(\delta) = g\left(R^k \boxplus \delta\right)$. For $y = R^k \boxplus \delta$ we can approximate

$$g(y) = g\left(R^k \boxplus \delta\right) = f(\delta) \approx f(0) + Jf(0)\,\delta = g\left(R^k\right) + J_\delta g\left(R^k \boxplus \delta\right)\big|_{\delta=0}\delta \tag{2.19}$$

where we linearized f at the origin. For the Jacobian matrix we need again the chain rule

$$Jg\left(R^k \boxplus \delta\right)\big|_{\delta=0} = J_R g\left(R^k\right) J_{\mathrm{w}} e^{\hat{\mathrm{w}}}(0) \tag{2.20}$$

When we now obtain an increment δ the update reads $\mathrm{w}^{k+1} = \log\left(R^k \boxplus \delta\right)$ if we use the representation in the Lie algebra as the optimization variable. If we use the representation in the Lie group the update is $R^{k+1} = R^k \boxplus \delta$.

The advantage of doing the increment in the Lie algebra is that we can do a direct update of w. The advantage of the Lie group is that we need to evaluate the derivative of Rodrigues' formula only at 0, which yields simple matrices that do not require any computation. In this thesis, we use the increment in the Lie algebra.

2.4 Warping

We now combine the camera model with the camera pose to obtain the mapping from one camera frame to another one. Consider Figure 9. For a general image point ${}_i\mathbf{x}$ in image i assume we are given the depth value ${}_ih$ of the corresponding 3D point $\mathbf{X}$ in the local coordinate system $X_iY_iZ_i$. In our case of digital images we read this value from a given depth matrix ${}_i\mathbf{h}$ in the reference frame, i.e.

$${}_ih = {}_i\mathbf{h}\left({}_i\mathbf{x}\right) \tag{2.21}$$

Also, assume that we are given the pose of camera j relative to camera i. Depending on those two quantities, we want to find the coordinates ${}_j\mathbf{x}$ of the projection of $\mathbf{X}$ in image j.

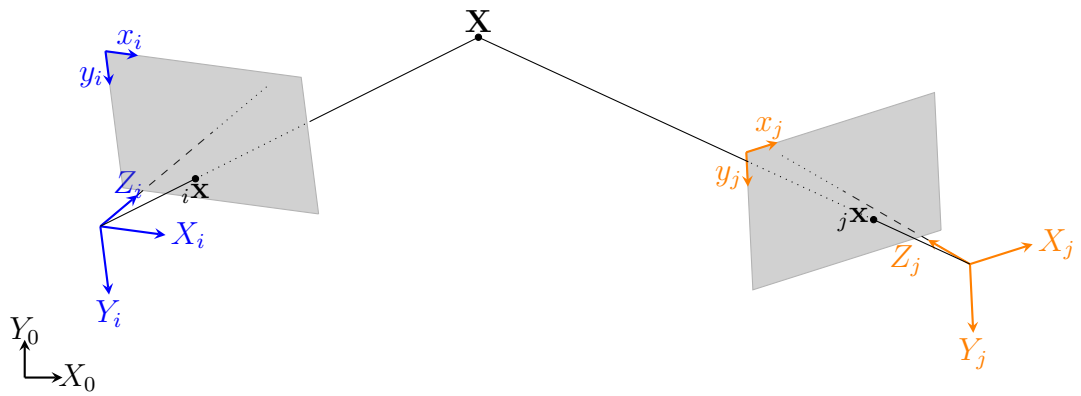


Figure 9: *The general point* $\mathbf{X}$ *observed in the camera frames* i *and* j*. The warping function* $\omega_{i\mathbf{x}}$ *maps the the image point* $_i\mathbf{x}$ *in the first image to the point* $_j\mathbf{x}$ *in the second image coordinate system, given the relative pose of* j *w.r.t.* i *and the depth value* $_i\mathbf{h}\,(_i\mathbf{x})$*.*

We start by un-projecting the image point $_i\mathbf{x}$ to obtain the corresponding 3D point in coordinates of system i. This is done using Equation (2.7).

$$_i\mathbf{X} = {}_ihK^{-1}{}_i\tilde{\mathbf{x}} \tag{2.22}$$

Using the coordinate transformation of Equation (2.11) we can represent the point $\mathbf{X}$ in coordinate system j.

$$_j\mathbf{X} = R_{ji}\,{}_i\mathbf{X} + T_{ji} = {}_ihR_{ji}K^{-1}{}_i\tilde{\mathbf{x}} + T_{ji} \tag{2.23}$$

We now project this representation of the 3D point into the image coordinate system j using again Equation (2.7). For readability, we omit the indices of the pose variables. Also, note that we do not explicitly show the dependence of the rotation matrix on its parameterization.

$${}_jh_j\tilde{\mathbf{x}} = K\,{}_j\mathbf{X} = {}_ihKRK^{-1}{}_i\tilde{\mathbf{x}} + KT \tag{2.24}$$

In Equation (2.24), we still have the unknown depth $_jh$ of the 3D point in the second coordinate system. However, as $_j\tilde{\mathbf{x}}$ is given in homogeneous coordinates we see, that the third component of the vector on the right in Equation (2.24) is precisely the unknown depth. We can, therefore, eliminate it by dividing the first two rows by the third one. This yields the warping equation, that, given the relative pose and the depth map, defines the mapping from one camera frame to another one.

$${}_j\mathbf{x} = \omega_{i\mathbf{x}}\,(R, T, {}_i\mathbf{h}) = \frac{1}{({}_ihKRK^{-1}{}_i\tilde{\mathbf{x}} + KT)_3}\begin{bmatrix}({}_ihKRK^{-1}{}_i\tilde{\mathbf{x}} + KT)_1\\ ({}_ihKRK^{-1}{}_i\tilde{\mathbf{x}} + KT)_2\end{bmatrix} \tag{2.25}$$

Note that we write the point $_i\mathbf{x}$ as a subscript of the warping function rather than as an argument. This keeps the notation clean when we calculate the derivative of ω w.r.t the pose and the depth map later. Moreover, note that the warping is smooth if the denominator is nonzero. Recall that the denominator is the depth of the 3D world point in the coordinate system of the second camera. Therefore, the case when the denominator is zero corresponds to a depth value of 0 in the image j,

i.e. the second camera being exactly on the point, which is somewhat pathological and can be ignored. In general, we need to treat points as invalid for which the denominator is smaller than zero as this corresponds to a negative depth in the second camera frame i.e. the point being behind the camera.

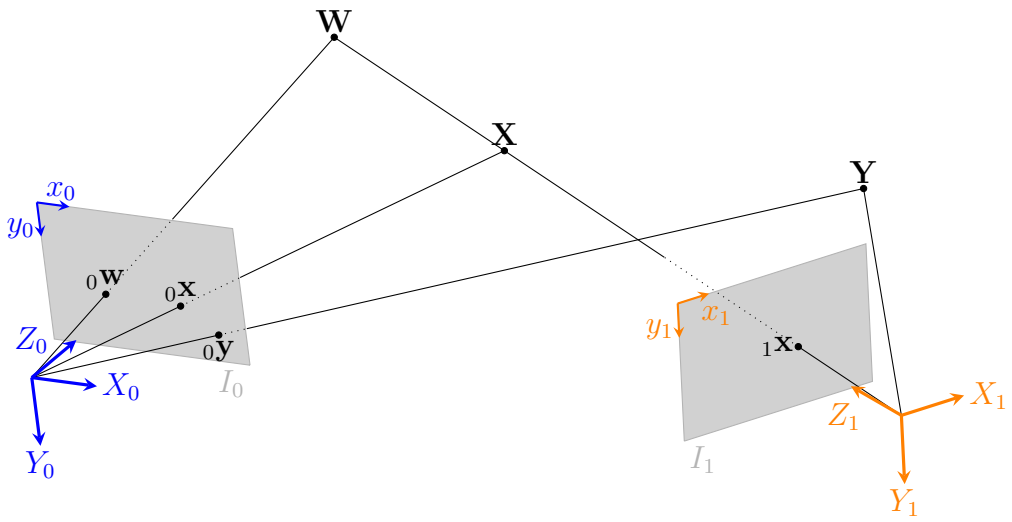


Figure 10: *The SLAM setting with two available camera frames - the reference frame I_0 in blue and the second frame I_1 in orange. We want to estimate a depth value for each pixel of the reference frame and the relative pose of the second frame. For all valid 3D world points like* $\mathbf{X}$*, the data term measures the difference in the image intensities at* $_0\mathbf{x}$ *and* $_1\mathbf{x}$*. Examples for invalid points are the point* $\mathbf{Y}$ *because it is not visible in the second frame and the point* $\mathbf{W}$ *because it is occluded by* $\mathbf{X}$*.*

3 Cost Function for the SLAM Approach

Using the model for the image formation process from the last chapter, we now state the SLAM problem more formally. As input data, we are given several images of the same static scene taken from different angles. We define one of the images to be the reference frame I_0 and denote the set of the indices of all other images by $\mathcal{I}$. For the reference frame, we want to estimate a depth map $\mathbf{h}$ that assigns each pixel a depth value.

Besides the depth map, we want to estimate the poses of the remaining camera frames relative to I_0. While for some applications, we might be interested in the poses because they describe the movement of the camera, we will see that we need the poses anyhow to be able to use the warping function for the photometric data term. The poses are represented by the exponential coordinates w_i of the rotation matrices and by the translation vectors T_i where the index i means that this quantity describes the pose of the i-th camera frame relative to the reference frame.

For the estimation of the depth map and the poses, we set up a cost function evaluating how good the parameters fit the data contained in the images. We define a photometric data term that compares the image intensities of the reference image to the intensities of one or more other images taken from different angles. As the purely data term based optimization turns out to be ill-posed, we introduce an additional regularization term. At the end of this chapter, we can state a non-linear, non-convex cost function for the estimation of the poses and the depth map.

3.1 Photometric Dataterm

We start with the more straightforward case of only two images of a scene taken from different positions, as shown in Figure 10. We define one camera coordinate

system to be the reference system $X_0Y_0Z_0$. All estimated poses are relative to this coordinate system, and the estimated depth map is given in this coordinate system. This is an arbitrary choice, and we could choose any other system as the global world reference system.

The idea behind a photometric data term is as follows. Consider an arbitrary pixel ${}_0\mathbf{x}$ in the reference image with intensity $I({}_0\mathbf{x})$. Depending on the pose of the second camera and the depth map in the reference frame, we warp the location of that pixel center into the second image and measure the intensity difference between $I({}_0\mathbf{x})$ and the intensity at the warped location. By doing this for all pixels in the reference image and summing over the intensity differences, we obtain a total data cost function. Then we try to find the pose and depth map that minimizes this cost.

The approach is called *photometric* because it measures the difference between image intensities. This is in contrast to geometric approaches like Bundle Adjustment, where a geometric error is minimized, see e.g. [43]. Before we can write down the data term, we need to address a few aspects.

3.1.1 Lambertian Surfaces

The key to justifying the photometric approach is to assume so-called *Labertian surfaces*. A surface is considered Lambertian if it does not change appearance when changing the angle of view. This requires the incoming light to be reflected uniformly in all directions. Matte surfaces come very close to the Lambertian model, while shiny surfaces like metals and mirrors are examples for non-Lambertian surfaces [28, Chapter 3.A]. While this assumption will rarely be fulfilled completely, it still provides a very good foundation for the data term and has been used in several photometric approaches, see e.g. [32], [42], [16] or [17]. However, we need to keep this assumption in mind as a source of errors, especially if we are dealing with wide camera baselines, changes in illumination, or if reflective surfaces are present.

3.1.2 Invalid points

It may happen that a 3D world point that is visible in the reference frame is not visible in the second frame. For an example consider point Y in Figure 10. While it is visible in the camera frame 0, it is outside of the field of view of camera 1. As we cannot compute an intensity difference for the point ${}_0\mathbf{y}$, we consider this point to be invalid. For the computation of the total data cost, we need to check for all points if they are valid so that we only use meaningful intensity differences in the cost function. We call the set of valid pixel centers $\mathcal{V}_{I_1} \subset \mathcal{P}_0$ where the subscript means that the pixels are valid with respect to image 1. This notation is helpful when we consider more than two images because a point that is visible in the second image might not be visible in a third one.

Another situation that might occur is that two points in the reference image map to the same point in the second image. Consider the points ${}_0\mathbf{x}$ and ${}_0\mathbf{w}$ corresponding to the 3D world points $\mathbf{X}$ and $\mathbf{W}$ in Figure 10. Although the warping equation maps both image points from I_0 to the same location in I_1 in the actual physical image, the point $\mathbf{W}$ is occluded, and only the point $\mathbf{X}$ is visible.

A naive approach would be to declare both points invalid. This method has advantages when the object that the point $\mathbf{W}$ is on is only partly occluded because $\mathbf{X}$ is on the edge of an object in the foreground. In that case, the pixels surrounding

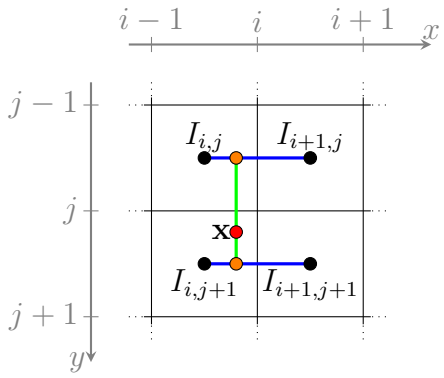


Figure 11: *Bilinear interpolation in the pixel grid. The interpolation at the point x can be seen as interpolation along the x direction (in blue) followed by interpolation of the results at the orange points in y direction (in green).*

${}_1\mathbf{x}$ did receive light emitted from both objects, and consequently, the intensities are a mixture that might be very different from the intensity at ${}_0\mathbf{x}$.

If the point $\mathbf{W}$ is fully occluded, however, considering both points as invalid means rejecting information as the point ${}_0\mathbf{x}$ is, in fact, a valid point. Therefore a second approach is used frequently. Whenever we detect an occlusion by detecting that two points in the reference image have been warped to the same location in the second image, we consider the corresponding depths in the second image and add the point with the smallest depth to the set of valid points while discarding the other one. Note that this approach adds some complexity and thus might not be computationally feasible.

Since the points are never mapped to the exact same location, we need to employ some proximity measure for both approaches for the occlusion detection.

3.1.3 Interpolation in the Second Image

To compute the photometric difference, we need the intensity at the warped location in the second image. This value could be obtained using the intensity mapping in Definition 2.1, i.e. we simply use the intensity value of the pixel in which the warped point is located. However, this mapping is discontinuous, which is conveyed to the cost function. As discontinuous cost functions are very hard to work with, we interpolate the brightness values at the pixel centers to obtain an intensity map that is at least continuous.

Although the interpolation is an approximation, we have to keep in mind that the piecewise constant image we are approximating is not the "correct" value either. It is itself a subsampling of the real intensity map that can be observed on the imaging surface. As this true intensity map is at least piecewise continuous, the interpolation can be seen as an inversion of the subsampling to the pixel level, which is why it is a reasonable approximation that proves reliable.

On an abstract level the interpolation $\mathcal{J}$ maps the intensity matrix $I \in \mathbb{R}^{m\times n}$ to a function $\mathcal{J}I \in C^k(\Omega)$ where we require $k \geq 0$. We will now state two common interpolation methods.

Bilinear Interpolation One of the easiest approaches to obtain a C^0 intensity map is to use bilinear interpolation. Consider Figure 11. The arbitrary point $\mathbf{x}$

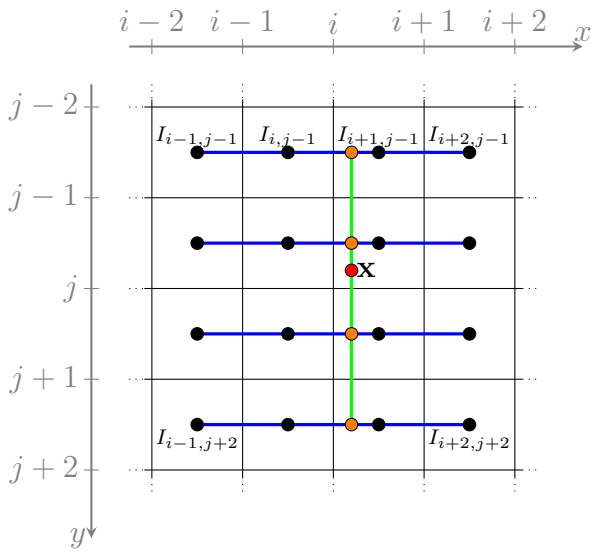


Figure 12: *Bicubic interpolation in the pixel grid. First a convolution with W is performed along the y direction (in blue). The results given at the orange points are then convoluted with the same kernel along the x direction.*

is located between the pixels (i, j), $(i+1, j)$, $(i, j+1)$ and $(i+1, j+1)$ and we want to interpolate the image intensity that is available at the pixel centers. The interpolation equation reads

$$\begin{aligned}\mathcal{J}_{lin} I(\mathbf{x}) = &(y_{j+1} - y)\left[(x - x_i) I_{i+1,j} + (x_{i+1} - x) I_{i,j}\right] \\ &+ (y - y_i)\left[(x - x_i) I_{i+1,j+1} + (x_{i+1} - x) I_{i,j+1}\right]\end{aligned} \tag{3.1}$$

where $(x_i, y_j) \in \mathcal{P}$ are the coordinates of the center of the pixel (i, j). The terms in the square brackets can be seen as one dimensional linear interpolations in x direction, as shown in blue in Figure 11 which is followed by a one dimensional interpolation of those two results in y direction, as shown in green.

Bicubic Convolution For some optimization methods we require the cost function to be at least C^1 which can be achieved by a bicubic approach. Keys [23] proposed an efficient convolution approach in which the interpolated value is obtained by convolution in both x and y direction with the kernel

$$W(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & \text{for } |x| \leq 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & \text{for } 1 < |x| < 2 \\ 0 & \text{else} \end{cases} \tag{3.2}$$

for $a = -0.5$.

A general 1 dimensional convolution f of a quantity p sampled at points t_i with a kernel K is given by

$$f(x) = \sum_i p(t_i) K(x - t_i) \tag{3.3}$$

For the bicubic convolution in the image consider Figure 12. As we use a separable kernel (convolution in each direction independently) we are first convolving in x direction along the blue lines. Afterwards, we convolve the results given at the

orange points in the y direction along the green line to obtain the interpolated value. The result reads

$$\mathcal{J}_{cub} I(\mathbf{x}) = \sum_l \sum_k I_{kl} W(x - x_k) W(y - y_k) \tag{3.4}$$

where again $(x_i, y_j) \in \mathcal{P}$ are the coordinates of the center of the pixel (i, j).

3.1.4 Dataterm for Two Images

We now define the data term in the case of two images, where we restrict ourselves to gray value images, i.e. images with only one channel. More channels are possible, but usually, the additional information does not compensate for the increased computational load. To measure the intensity difference, we can use an arbitrary loss function ℓ. We discuss different choices for the loss function in Section 3.2.

The dataterm for two available images reads

$$E_{data}(\mathrm{w}, T, \mathbf{h}) = \sum_{\mathbf{x} \in \mathcal{V}_{I_1}} \ell\left(\mathcal{J} I_1\left(\omega_{\mathbf{x}}(R(\mathrm{w}), T, \mathbf{h})\right) - I_0(\mathbf{x})\right) \tag{3.5}$$

Here $\omega_{\mathbf{x}}(R(\mathrm{w}), T, h)$ denotes the warping of the pixel center $\mathbf{x}$ with the associated depth $h = \mathbf{h}(\mathbf{x})$ from the reference image to image 1. As it is clear from the first argument of the warping function which depth we are referring to, we shorten the notation for the depth to keep the equation as clean as possible.

R and T denote the relative pose of camera 1 with the exponential coordinates w. We omit the indices of the coordinate systems because the direction of the warping is apparent in the case of two images. If we are using more than one channel, we need to evaluate the loss function for the difference in each channel.

Dataterm for multiple images

We can easily extend this dataterm to multiple images. We simply add the dataterms obtained by comparing the intensities of the individual images to the reference. Recall that $\mathcal{I}$ denotes the indices of all available images without the reference image. Then the dataterm for multiple images reads

$$E_{data}(\mathbf{w}, \mathbf{T}, \mathbf{h}) = \sum_{i \in \mathcal{I}} \sum_{\mathbf{x} \in \mathcal{V}_{I_i}} \ell\left(\mathcal{J} I_i\left(\omega_{\mathbf{x}}(R(\mathrm{w}_i), T_i, \mathbf{h})\right) - I_0(\mathbf{x})\right) \tag{3.6}$$

where $\mathbf{w}$ and $\mathbf{T}$ denote the stacked exponential coordinates and translation vectors of all available camera frames, i.e.

$$\mathbf{w} = \begin{bmatrix} \mathrm{w}_1 \\ \vdots \\ \mathrm{w}_{|\mathcal{I}|} \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} T_1 \\ \vdots \\ T_{|\mathcal{I}|} \end{bmatrix}$$

Note that now the set of valid points $\mathcal{V}_{I_i}$ must be determined for each individual image. Additionally, since it depends on $\mathbf{w}, \mathbf{T}$ and $\mathbf{h}$, it must be updated with the optimization variables in iterative optimization methods. Depending on the chosen validity approach, this might require some computational effort.

3.2 Loss Functions

A loss function (or cost function) is usually used to set up an optimization problem for a parameter estimation where we want to find the best parameters for a model given a set of observations. The loss function assigns a real value (cost) to a given parameter hypothesis to evaluate how good the model fits the data for this hypothesis. Often the different data points can be treated individually, and we sum the losses for each observation to obtain a total cost. Depending on the context, the estimation problem is either written as a minimization or a maximization. To obtain well-posed problems, the loss functions must be lower or upper bound, respectively.

Depending on the problem at hand, we distinguish between loss functions for classification and regression. In classification, we want to obtain a model to categorize data, and therefore the loss function is constructed to be an indicator that shows if the categorization of a sample with known category is correct or not. Due to the categories, this is a discrete setting. In regression, on the other hand, we try to fit a functional dependence to the observed data, and therefore the loss function measures how good the current parameter hypothesis fits the observed data in a more continuous sense.

For this thesis, it is clear that we are in the setting of regression. The parameters to estimate are the pose and the depth map, the model is the warping followed by the interpolation of the brightness, and the available data we compare the model to is the brightness in the reference image. Therefore we focus on the regression case only and do not discuss classification loss functions in this thesis.

In the following, we introduce 3 popular choices of regression loss functions for minimization problems. In general, we have a model f parameterized by a vector θ and a set of data points (x_i, y_i) where the x_i are the inputs and the y_i the outputs of the model. We are interested in regression loss functions ℓ that assign a positive value to the difference between the prediction of the model and the observed data. We sum the loss for each available data point to obtain an overall cost function

$$E(\theta) = \sum_i \ell\left(y_i - f(x_i, \theta)\right) \tag{3.7}$$

By minimizing this cost function, we obtain the optimal parameter θ.

Quadratic Loss

This is the classical choice resulting in the least-squares approach. The loss function reads

$$\ell_2(x) = \frac{1}{2}x^2 \tag{3.8}$$

and is shown in blue in Figure 13. It is immediate that this cost function is differentiable, which allows using methods from smooth optimization. The downside of this loss is that it is not robust against outliers. As seen in the plot, larger differences are penalized drastically due to the quadratic growth of the function. This means that erroneous measurements have a huge impact on the optimization.

Absolute Loss

A robust loss function is given by the absolute loss

$$\ell_1(x) = |x| \tag{3.9}$$

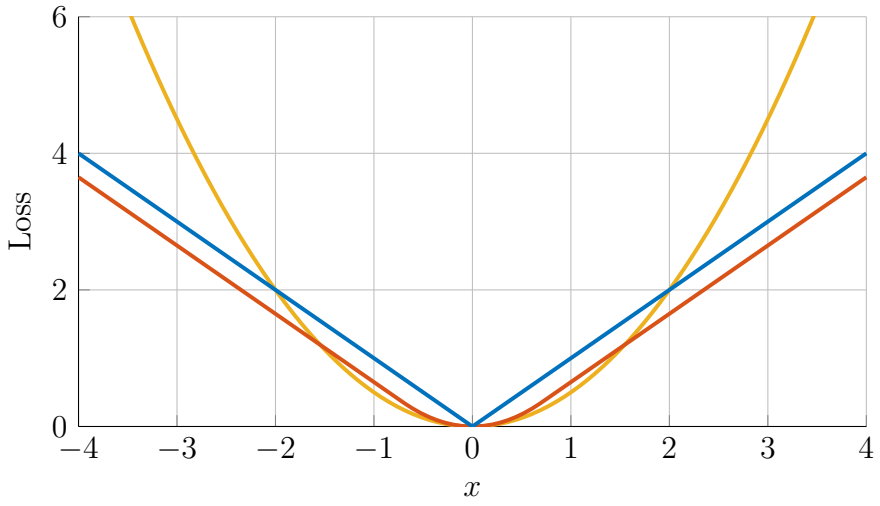


Figure 13: *Comparison of different regression loss functions $\ell(x)$. The quadratic loss is shown in yellow, the absolute loss in blue and the Huber loss with Huber parameter $h = 0.7$ in red.*

which is shown in red in Figure 13. Due to the linear growth, outliers have much less impact on the optimization, which is the reason for the robustness. The disadvantage of this loss is that it is not differentiable any more, and therefore non-smooth optimization techniques need to be applied.

Also, we see that the slope is constant, even as we are approaching the minimum. That means that when we apply an iterative optimization method, we cannot base the step width on this slope alone as we would still make huge steps near the minimum, which results in overshooting. Instead, we need to apply a step width rule since there is no natural step width reduction. Note the contrast to the quadratic loss where the slope goes to zero as we are approaching the minimum.

Huber Loss

The Huber loss combines the advantages of both the quadratic and the absolute loss. For deviations smaller than a threshold h it is quadratic and for bigger deviations it transitions to a linear growth. This is achieved by the following loss function

$$\ell_h(x) = |x|_h = \begin{cases} \frac{1}{2h}x^2 & \text{if } |x| \leq h \\ |x| - \frac{h}{2} & \text{else} \end{cases} \tag{3.10}$$

which is shown in yellow in Figure 13. The result is a differentiable loss function with decreasing gradient towards the minimum, that still has the robustness properties of the absolute loss.

A Probabilistic View on Regression

In the following we will shortly discuss regression from a probabilistic point of view. This will provide a different motivation for the loss functions. Assume that the data points (x_i, y_i) are independent and that we can model the errors e.g. due to noise by an additive normal distribution with zero mean and constant variance σ^2. Then we can write

$$y_i = f(x_i, \theta) + \varepsilon_i \tag{3.11}$$

with $\varepsilon_i \sim \mathcal{N}(0, \sigma^2)$. From the transformation rule it follows that the y_i are again normally distributed with

$$y_i \overset{iid}{\sim} \mathcal{N}\left(f(x_i, \theta), \sigma^2\right) \tag{3.12}$$

We use a maximum likelihood approach to estimate the optimal parameter vector θ. Recall that the pdf of the Gaussian distribution reads $f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}$. Therefore, due to the independence, the likelihood function can be written as

$$\begin{aligned} L\left(\theta, \sigma^2; \mathbf{y}\right) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{\left(y_i - f(x_i, \theta)\right)^2}{2\sigma^2}\right\} \\ &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left\{-\frac{\sum_{i=1}^{n}\left(y_i - f(x_i, \theta)\right)^2}{2\sigma^2}\right\} \end{aligned} \tag{3.13}$$

Instead of maximizing the likelihood function we can maximize the log-likelihood function

$$l\left(\theta, \sigma^2; \mathbf{y}\right) = \ln\left(L\left(\theta, \sigma^2; \mathbf{y}\right)\right) = -\frac{n}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}\left(y_i - f(x_i, \theta)\right)^2 \tag{3.14}$$

The maximization of the log-likelihood function is equivalent to the minimization

$$\min_{\theta}\left\{-\ln\left(L\left(\theta, \sigma^2; \mathbf{y}\right)\right)\right\} \tag{3.15}$$

which in case of the quadratic loss becomes

$$\min_{\theta} \frac{1}{2}\sum_{i=1}^{n}\left(y_i - f(x_i, \theta)\right)^2 \tag{3.16}$$

This is exactly the regression problem with the quadratic loss.

We obtain the absolute loss function if we model the error as iid Laplace distributed. The pdf of the Laplace distribution reads $f(x) = \frac{1}{2b}\exp\left\{-\frac{|x-\mu|}{b}\right\}$ with the parameters μ and b. Analogously to before we construct the likelihood function

$$L\left(\theta, b; \mathbf{y}\right) = \left(\frac{1}{2b}\right)^n \exp\left\{-\frac{\sum_{i=1}^{n}\left|y_i - f(x_i, \theta)\right|}{b}\right\} \tag{3.17}$$

and obtain

$$\min_{\theta}\sum_{i=1}^{n}\left|y_i - f(x_i, \theta)\right| \tag{3.18}$$

which is the regression with the absolute loss.

An intuition of why the absolute loss is more robust than the quadratic loss can be obtained by comparing the normal distribution to the Laplace distribution. For large deviations from the mean, it holds that $(x-\mu)^2 > |x-\mu|$, which means that the value of the pdf of the normal distribution is smaller than the pdf of the Laplace distribution. We see that the Laplace distribution has "heavier tails", i.e. large deviations are more likely in this model. This is the reason why the absolute loss function is better suited to deal with outliers.

We can see the Huber loss regression as the solution strategy to a likelihood maximization similar to before, where we now model the error as a combination of both distributions. For smaller deviations, the error is distributed normally, while for larger deviations, the error follows the Laplacian distribution.

3.3 Regularization

In most cases, we are not able to recover satisfactory estimates for the pose and the depth map from the data term alone. For any pixel in the reference image, there might be multiple points in the second image that have a very similar interpolated intensity value and that we can warp to by using different pose and depth combinations. An easy example is a surface with a uniform color. Varying the depth hypothesis for points that are associated with that surface does not result in significant changes in the data cost, at least locally.

Moreover, if we have found multiple points in the second image with similar intensities, the correct point is not necessarily among them as we have seen that assuming Lambertian surfaces is an idealization that does not always hold. This makes the estimation very sensitive to external influences during the image formation process. A third problem arises for invalid points. Since the data term does not depend on the depth for those pixels, we have no information on how to improve this depth value.

As a result, besides being unstable, estimation based solely on the data term results in a very noisy depth map. A common approach to face this problem is to use a regularization term E_{reg} that favors smoothness of the depth map. The idea is that, apart from discontinuities at object boundaries, the depth varies smoothly along object surfaces.

3.3.1 Different Regularization Terms

We use forward differences as an approximation of the first derivative to measure the smoothness of the depth map at each pixel. We choose the pixel centers as evaluation points.

Definition 3.1 (Discrete Gradient Operator Forward Differences)**.** Let X denote the Euclidean space $\mathbb{R}^{m\times n}$, let $\mathbf{h} \in X$ be a matrix and let $\mathbf{x} = (x, y) \in \mathcal{P}$ be the coordinates of a pixel center. Using the indexing from Definition 2.1 we define the discrete gradient operator $D : X \to X \times X$ by

$$D\mathbf{h}(\mathbf{x}) = \begin{bmatrix} D_x\mathbf{h}(\mathbf{x}) \\ D_y\mathbf{h}(\mathbf{x}) \end{bmatrix}$$

with the forward difference operators defined as

$$D_x\mathbf{h}(\mathbf{x}) = \begin{cases} \mathbf{h}(x+1, y) - \mathbf{h}(x, y) & \text{if } x < n \\ 0 & \text{else} \end{cases}$$

$$D_y\mathbf{h}(\mathbf{x}) = \begin{cases} \mathbf{h}(x, y+1) - \mathbf{h}(x, y) & \text{if } y < m \\ 0 & \text{else} \end{cases}$$

Note that D is a linear operator.

Using the discrete gradient operator, we now discuss several different approaches to construct a regularization term.

Discrete Total Variation (TV) Regularization

A classical choice is to use a discrete version of the total variation. In this case we sum the Euclidean norms of the gradients for all pixel, and therefore the regularization term reads

$$E_{reg}(\mathbf{h}) = \sum_{\mathbf{x}\in\mathcal{P}} \|D\mathbf{h}(\mathbf{x})\|_2 \tag{3.19}$$

Note that we can see this as the L^1 norm of the vector containing the L^2 norms of the gradient at each point. This smoothing term was first proposed by Rudin, Osher and Fatemi [41] and has since been used countless times for different aspects of computer vision, like range image integration [47], optical flow estimation [45] or as in our case for SLAM [42].

Remark. Many authors prefer to state energy-based methods like ours in a continuous setting. Most of the mathematical analysis, like proofs for existence or uniqueness, is more natural in this setting as we can employ results from the theory of partial differential equations or the calculus of variations. Equation (3.19) is, in fact, the discretization of the standard total variation that is defined in the continuous setting. As the focus of this thesis is on the implementation aspects of SLAM, we do not discuss the continuous setting here, which requires an extensive foundation from functional analysis. We refer to the book by Aubert and Kornprobst for a rigorous discussion of the continuous case [1].

Huber Regularization

The TV regularization preserves jumps in the depth values at object boundaries due to the L^1 norm. However, as the sparsity of the gradients is favored, the approach leads to piecewise constant depths for surfaces, which is known as staircasing effect. Werlberger et al. observed this effect for the regularization of a flow field in an optical flow problem and proposed the robust Huber norm instead of the Euclidean norm to avoid it [46]. The Huber norm combines the squared Euclidean norm for small gradient magnitudes with the standard Euclidean norm for larger magnitudes and can be seen as a generalization of the Huber loss defined previously to higher dimensions.

Definition 3.2 (Huber Norm)**.** Let $h > 0$ be the Huber parameter. Then the Huber norm of $x \in \mathbb{R}^n$ is defined as

$$\|x\|_h = \begin{cases} \frac{1}{2h}\|x\|_2^2 & \text{if } \|x\|_2 \leq h \\ \|x\|_2 - \frac{h}{2} & \text{else} \end{cases}$$

The quadratic term corresponds to a down-weighting of vectors with a small norm as $\|x\|_2^2 < \|x\|_2$ for $\|x\|_2 < 1$. For the 1 dimensional case, the effect can be seen in Figure 13 (note that the Euclidean norm is the absolute value in this case). For small values, the quadratic terms are considerably smaller than the linear term. In higher dimensions, similar behavior can be observed. Intuitively, when using the Euclidean norm, small costs are only possible very close to zero gradient, which explains the staircasing effect. In contrast, for the Huber norm, the smaller costs for small gradients allow the algorithm more flexibility in those regions, which improves the smoothness of the result.

The TV Huber regularization term is defined analogously to the TV regularization and reads

$$E_{reg}(\mathbf{h}) = \sum_{\mathbf{x}\in\mathcal{P}} \|D\mathbf{h}(\mathbf{x})\|_h \tag{3.20}$$

This regularization term is used e.g. in [25] and in DTAM [32].

Total Generalized Variation

The total generalized variation (TGV) is a generalization to the TV regularization and was introduced and analyzed rigorously for the continuous setting by Bredies et al. [5]. It allows additional regularization of higher-order derivatives. While, in principle, any order of derivatives is possible, we only discuss the second-order case here. It is well suited for affine surfaces like walls and many other man-made objects.

The TGV regularization of second order can be written as

$$E_{reg}(\mathbf{h}) = \alpha_1 \sum_{\mathbf{x}\in\mathcal{P}} \|D\mathbf{h}(\mathbf{x}) - \mathbf{v}(\mathbf{x})\|_1 + \alpha_0 \sum_{\mathbf{x}\in\mathcal{P}} \|\mathcal{E}(\mathbf{v})(\mathbf{x})\|_1 \tag{3.21}$$

where the newly introduced $\mathbf{v} \in \mathbb{R}^{m\times n\times 2}$ is added to the optimization variables. In terms of indexing we treat $\mathbf{v}$ as a 2-channel image. The positive weights α_0 and α_1 are used to balance the terms. In the original work of Bredies et al. $\mathcal{E}$ is the symmetrized Jacobian which, assuming a continuous vector field $\mathbf{v}^{cont}$ reads

$$\mathcal{E}\left(\mathbf{v}^{cont}\right)(\mathbf{x}) = \frac{1}{2}\left(J\mathbf{v}^{cont}(\mathbf{x}) + J\mathbf{v}^{cont}(\mathbf{x})^T\right) \tag{3.22}$$

The discretization for our continuous setting is given by

$$\mathcal{E}(\mathbf{v})(\mathbf{x}) = \begin{bmatrix} D_x\mathbf{v}_1(\mathbf{x}) & \frac{D_y\mathbf{v}_1(\mathbf{x})+D_x\mathbf{v}_2(\mathbf{x})}{2} \\ \frac{D_y\mathbf{v}_1(\mathbf{x})+D_x\mathbf{v}_2(\mathbf{x})}{2} & D_y\mathbf{v}_2(\mathbf{x}) \end{bmatrix} \tag{3.23}$$

Note that this is still a linear operator.

In the formulation of the regularization term, the L^1 norm means the sum of the absolute values of the entries. The TGV regularization using the symmetrized Jacobian is used i.e. in [37]. It is, however, also possible to use the standard Jacobian instead of the symmetrized version, which was done in [25].

To finish the section on TGV regularization, we want to build an intuition on how the second-order case works. In contrast to standard TV, we subtract a vector field $\mathbf{v}$ before penalizing the gradient of the depth map. This vector field itself is required to have low variation to obtain minimal cost. Now consider an affine surface. The gradient of its depth map is constant. If we choose the vector field $\mathbf{v}$ to be this gradient, we achieve zero cost in the first term, but also in the second term because the variation of a constant is zero as well. This is why the second-order TGV regularization favors affine surfaces.

Isotropic and Anisotropic approaches

In all the regularization methods discussed so far, we sum the magnitudes of the gradient measured in some norm for all points. The norm introduces a coupling

of the derivative in x and y direction. This coupling can complicate minimization algorithms substantially as it affects separability of the cost function. Therefore we sometimes resolve this coupling by measuring the derivative in each direction independently. In contrast to the standard approach described before which is considered *isotropic*, the decoupling is referred to as *anisotropic*. An anisotropic version of the standard TV regularization therefore reads

$$E_{reg}(\mathbf{h}) = \sum_{\mathbf{x}\in\mathcal{P}} \|D_x\mathbf{h}(x)\|_2 + \|D_y\mathbf{h}(x)\|_2 = \sum_{\mathbf{x}\in\mathcal{P}} |D_x\mathbf{h}(x)| + |D_y\mathbf{h}(x)| \tag{3.24}$$

Note that $D_x\mathbf{h}(x)$ and $D_y\mathbf{h}(x)$ are the scalar directional derivatives at $\mathbf{x}$, and therefore the Euclidean norm is just the absolute value. Anisotropic versions of the other approaches can be constructed analogously.

In the context of the regularization of images, the term *isotropic* means rotation invariant, which means that the value of the regularization energy does not change if we rotate the image coordinate system. However, as discussed by Condat [9] true isotropy is a property of the total variation only in the continuous case. Due to the approximation of the gradient by finite differences, the property is lost for discrete images. Therefore, the "isotropic" discrete total variation is only an approximation yielding reasonably good results. While Condat proposes an improved version of the total variation, in this thesis, we stick to the formulations introduced in the previous sections for simplicity.

3.3.2 Image-Driven Adaptive Regularization Weights

To improve the sharpness of the discontinuities of the depth map at image boundary, we adopt an idea that is used in DTAM [32]. The idea is, that discontinuities in the depth map often coincide with discontinuities in the image brightness, e.g. due to different colors of the object in the foreground and the background. Therefore, we weight the regularization for each pixel individually, depending on the intensity gradient in that pixel. The image-driven weight used in DTAM reads

$$\gamma(\mathbf{x}) = e^{-\alpha\|DI_0(\mathbf{x})\|_2^\beta} \tag{3.25}$$

where DI_0 is the image gradient in the reference image and α, and β are shape parameters.

While we can use this term for all the regularization terms introduced before, we will only give the combination with the Huber regularization as an example. The other terms are combined with the adaptive weight analogously. The adaptively weighted Huber regularization reads

$$E_{reg}(\mathbf{h}) = \sum_{\mathbf{x}\in\mathcal{P}} \gamma(\mathbf{x}) \|D\mathbf{h}(\mathbf{x})\|_h \tag{3.26}$$

with the function γ as defined in Equation (3.25).

3.3.3 A Probabilistic View on Regularization

We can also interpret regularization in a probabilistic sense. To do so, we need to employ the framework of Bayesian estimation. We conduct the discussion again in

the general setting, as we did in the section on the probabilistic view on regression. The most important step towards using the Bayesian theory is to consider the parameters θ not as fixed quantities but as a random variable that has a probability distribution.

We now use the observed data to update this distribution for θ. Assume that we have beforehand knowledge about the parameters, which is expressed as a *prior distribution* $f(\theta)$. This is our belief on what values for θ are likely "before seeing the data". Using Bayes rule, we can express the distribution of "θ after seeing the data $\mathbf{y}$" as

$$f(\theta \mid \mathbf{y}) \propto f(\mathbf{y} \mid \theta) f(\theta) \tag{3.27}$$

which we refer to as *posterior distribution*. The only unknown quantity is $f(\mathbf{y} \mid \theta)$, which corresponds to "probability of the data given θ". We see, that this is the standard likelihood that we used before, i.e.

$$f(\mathbf{y} \mid \theta) = L(\theta; \mathbf{y}) \tag{3.28}$$

which allows us to express the posterior distribution.

To be able to work with a numeric value for θ, rather than with a distribution, we maximize the posterior distribution to obtain an estimate for θ. This is only one possible choice, taking the mean is another popular approach. As before, we re-write the maximization as a minimization and use the logarithm to get rid of the product. The result reads

$$\min_{\theta} \{-\ln(f(\mathbf{y} \mid \theta)) - \ln(f(\theta))\} \tag{3.29}$$

Since $f(\mathbf{y} \mid \theta) = L(\theta; \mathbf{y})$, we see that the first term is exactly the regression loss term, which was derived in Section 3.2. For example, if we choose the Gaussian error model, the term becomes the quadratic loss. The second term can be seen as the additive regularization introduced in this chapter, however, now with a probabilistic interpretation. Due to the terminology introduced here, authors working in the Bayesian framework often refer to the "prior" instead of the "regularization term". We do not discuss prior distributions here, as this is beyond the scope of this thesis. We refer to the book by Chalmond [6].

3.4 The SLAM Optimization Problem

By combining the terms of the previous sections, we can now state the SLAM optimization problem. After introducing a weighting $\lambda_{reg} > 0$, to balance the data and the regularization term, the SLAM optimization problem reads as follows.

Problem 3.1. Let $I_0, \dots I_k$ be $k+1$ images of the scene, where I_0 is the reference image frame for which we estimate the depth map $\mathbf{h}$. Let $\mathcal{I} = \{1, \dots, k\}$ denote the indices of the additionally available images. Let w_i and T_i denote the (unknown) exponential coordinates and the translation of the i-th camera frame with respect to the reference frame. We refer to the stacked versions of them by $\mathbf{w}$ and $\mathbf{T}$, respectively.

Find $\mathbf{w}_{opt}$, $\mathbf{T}_{opt}$ and $\mathbf{h}_{opt}$ that solves

$$\min_{\mathbf{w},\mathbf{T},\mathbf{h}} \{E_{data}(\mathbf{w}, \mathbf{T}, \mathbf{h}) + \lambda_{reg} E_{reg}(\mathbf{h})\}$$

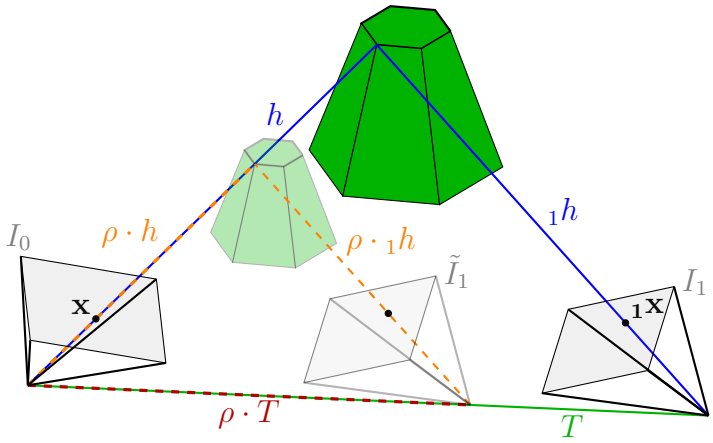


Figure 14: *Scale Freedom for monocular systems. If we scale the scene and the camera translation T by the same factor ρ, the image I_0 remains unchanged, and I_1 and $\tilde{I}_1$ are identical. Therefore, from the data in the images alone, a monocular system can only estimate the scene up to scale.*

Where we need to choose a loss function ℓ and an image interpolation method $\mathcal{J}$ for the dataterm

$$E_{data}\left(\mathbf{w}, \mathbf{T}, \mathbf{h}\right) = \sum_{i \in \mathcal{I}} \sum_{\mathbf{x} \in \mathcal{V}_{I_i}} \ell\left(\mathcal{J} I_i\left(\omega_{\mathbf{x}}\left(R\left(\mathrm{w}_i\right), T_i, \mathbf{h}\right)\right) - I_0\left(\mathbf{x}\right)\right)$$

and one of the regularization terms that were discussed in Section 3.3.1.

Recall that $\mathcal{V}_{I_i} \subset \mathcal{P}$ denotes the subset of all pixels in the reference image that, given the current pose and depth hypothesis, is visible in camera frame i.

Remark. The natural choice of parameterizing the depth map directly in the optimization variable might not be the optimal choice. Civera et al. proposed to use the inverse depth as parametrization for the SLAM problem [8]. In combination with the standard extended Kalman filter, they reported advantages for scenes with a depth map that covers a huge range of values.

3.5 Scale Freedom and Effects on the Cost Function

We now address a problem that is inherent to all monocular methods and prevents us from fixing the global length scale of the scene. For an intuitive understanding of the problem, consider Figure 14. Without any prior information on T or the depth map, we cannot know if the scene is the bigger cone for which the second image has been captured by camera I_1, or if the scene is the smaller cone captured by camera $\tilde{I}_1$. The image I_0 is the same in both cases, and the images I_1 and $\tilde{I}_1$ are likewise identical.

To see this quantitatively, we consider the warping in Equation (2.25) which we repeat here with adjusted indices.

$${}_1\mathbf{x} = \omega_{\mathbf{x}}\left(R, T, \mathbf{h}\right) = \frac{1}{\left(hKRK^{-1}\tilde{\mathbf{x}} + KT\right)_3} \begin{bmatrix} \left(hKRK^{-1}\tilde{\mathbf{x}} + KT\right)_1 \\ \left(hKRK^{-1}\tilde{\mathbf{x}} + KT\right)_2 \end{bmatrix} \tag{3.30}$$

The equation describes the warping of a general point $\mathbf{x}$ from image I_0 to its position ${}_1\mathbf{x}$ in the second image I_1, given the depth value h in the reference frame. Recall

that the denominator is the depth ${}_1h$ in the second camera coordinate system. We see that we can scale both the translation T and the depth value h by some constant ρ without changing the resulting position ${}_1\mathbf{x}$ in the second image.

From a theoretical standpoint, the consequences for our approach are severe. Without regularization, the scale freedom means that we can scale the depth map and the translation by the same constant ρ without changing the cost. This means that for a pure data term, we can only estimate up to scale.

We now consider the additional regularization term. For each term that we discussed, we can pull out the scaling constant ρ due to linearity and the properties of the norms, such that the constant scales the complete term. Therefore, since ρ does not influence the data term, we obtain zero cost by choosing $\rho = 0$. This corresponds to a zero depth map, no translation, and an arbitrary rotation, which is the global minimum of the cost function.

In practice, the problem turns out to be less severe. As we will see in the following sections, we employ iterative methods to minimize the cost function. The methods limit the progress on the optimization variable. Therefore, rather than big jumps, we expect a moderate drift of the scale, if we encounter any drift at all. Hence, if we end the optimization after sufficiently few iterations, we can recover the pose and the depth map on a scale close to the one of the initial values, and the effect of the scale drift is negligible. In Section 6.5, we investigate the influence of scale drift on our approach.

Note that the scale freedom also means that we are highly dependent on the initial values. If we initialize the algorithm with values on a wrong scale, we cannot hope to recover the right scale.

4 Optimization

The optimization problem stated in Problem 3.1 is non-linear and non-convex, which makes solving it quite challenging. Fortunately, with an additional approximation, we can identify the structure of the problem as a *convex-composite problem*, which allows us to use the *prox-linear algorithm*. The algorithm yields a sequence of strongly convex sub-problems that we solve using the *primal-dual hybrid gradient method*.

In this chapter, we discuss the methods in general, the application to the SLAM problem follows in Section 5.

4.1 The Proximal Mapping

For both algorithms, we need the proximal operator (or simply "prox operator"). We give the definition and an intuition on why this mapping is useful in the context of optimization. For thorough discussion we refer to [2, Chapter 6].

Definition 4.1 (Proximal Operator)**.** Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be a function that maps to the extended real line.[3]We define the proximal mapping with the step width parameter $\tau \in \mathbb{R}$ as

$$\operatorname{prox}_{\tau f}(u) := \underset{v \in \mathbb{R}^n}{\arg\min} \left\{ \tau f(v) + \frac{1}{2} \|v - u\|_2^2 \right\} \tag{4.1}$$

In general, the result of the proximal mapping can be a set. The next theorem gives a condition under which the result is unique.

Theorem 4.1 ([2, Theorem 6.3])**.** Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be proper, closed and convex and $\tau \in \mathbb{R}$. Then $\operatorname{prox}_{\tau f}(x)$ has a unique solution for all $x \in \mathbb{R}^n$.

Proof. The prox operator is defined by minimizing the function

$$g(v) = \tau f(v) + \frac{1}{2} \|v - u\|_2^2 \tag{4.2}$$

The first part of this function is convex by definition, and the second part is strictly convex, as the Hessian is the identity matrix. Since the sum of a convex and a strictly convex function is strictly convex again, we see that g is strictly convex. From the quadratic term, it is immediate that g is coercive. Together with the convexity, this shows the existence of a minimum. The strict convexity shows that the minimum is unique. □

For differentiable functions f, we can interpret the proximal mapping as an implicit gradient descent step, which is why the mapping is useful in the context of

[3]The extended real line is defined as $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. For $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ we define the domain $\operatorname{dom}(f) = \{x \in \mathbb{R}^n \mid f(x) < \infty\}$. We say f is *proper* if $\operatorname{dom}(f) \neq \emptyset$. By explicitly allowing infinite values we can e.g. conveniently construct a convex indicator function for a convex set C via $\delta_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{else} \end{cases}$

optimization. We see this by writing down the optimality conditions for Equation (4.1) if $f \in C^1$.

$$\begin{aligned} 0 &= \tau \nabla f(v^*) + v^* - u \\ v^* &= u - \tau \nabla f(v^*) \end{aligned} \tag{4.3}$$

This shows that the proximal mapping $\text{prox}_{\tau f}(u) = v^*$ performs an implicit gradient descent step with the step width τ. An ascent step can be done by using a negative stepwidth. By using subdifferentials as generalization of the gradient, this reasoning still holds for non-differentiable convex functions f. Hence, we can use the proximal mapping to construct gradient-like methods for non-differentiable functions.

The parameter τ controls how far we can go away from the point u. This becomes more obvious if we re-write the proximal mapping in the following equivalent form.

$$\text{prox}_{\tau f}(u) := \underset{v \in \mathbb{R}^n}{\arg\min} \left\{ f(v) + \frac{1}{2\tau} \|v - u\|_2^2 \right\} \tag{4.4}$$

The smaller τ the bigger the weighting of the quadratic term which means that we cannot go too far away from u. We will now extend this idea and assign each component of the optimization variable its own stepwidth to be able to control the progress of each component individually.

Definition 4.2 (Weighted Proximal Operator)**.** Let $f : \mathbb{R}^n \to \overline{\mathbb{R}}$ be closed, proper and convex and let $M \in \mathbb{R}^{n \times n}$ be a symmetric positive definite matrix. Then we define the weighted proximal mapping as

$$\text{prox}_{Mf}(u) := \underset{v \in \mathbb{R}^n}{\arg\min} \left\{ f(v) + \frac{1}{2} \|v - u\|_M^2 \right\}$$

where the weighted norm is defined as $\|u\|_M^2 = \langle M^{-1}u, u \rangle$. It is easy to show that $\|\cdot\|_M$ is indeed a norm.

While a general symmetric and positive definite matrix M allows incorporating cross dependencies between the components, an individual step width for each component can be obtained by a diagonal matrix. Note that the inverse of a positive definite matrix is still positive definite. Hence the weighted quadratic term remains strictly convex with Hessian M^{-1}, and therefore the weighted proximal mapping still yields a unique result which can be shown analogously to Theorem 4.1.

4.2 The Prox-Linear Algorithm

The prox-linear algorithm can be used for the minimization of the following class of problems

Definition 4.3 (composite optimization problem)**.** Let $g : \mathbb{R}^d \to \overline{\mathbb{R}}$ and $h : \mathbb{R}^m \to \mathbb{R}$ be proper, closed and convex functions and $c : \mathbb{R}^d \to \mathbb{R}^m$ be a C^1-smooth map. The we call the problem

$$\min_x F(x) := g(x) + h(c(x))$$

a composite optimization problem.

4.2.1 The Standard Algorithm

In the standard prox-linear algorithm we iteratively linearize the map c at the current point x^k and do a proximal step with stepwidth t[4] on the resulting convex function. Therefore the iterates are defined by the minimization problem

$$\begin{aligned} x^{k+1} &= \arg\min_{x\in\mathbb{R}^d} F_t\left(x; x^k\right) \\ &= \arg\min_{x\in\mathbb{R}^d} \left\{ g(x) + h\left(c\left(x^k\right) + Jc\left(x^k\right)\left(x - x^k\right)\right) + \frac{1}{2t}\left\|x - x^k\right\|^2 \right\} \end{aligned} \tag{4.5}$$

where Jc is the Jacobian matrix of c; see Appendix A for the definition used in this thesis.

An analysis of the algorithm can be found e.g. in the work of Drusvyatskiy et al. [15],[14]. The main step is to derive a condition under which F_t is an upper model, which means that it holds that $F_t\left(x; x^k\right) \geq F(x)$ for all $x, x^k \in \mathrm{dom}(g)$. We need to make the following assumptions

1. Let h be L-Lipschitz continuous, i.e.
$$|h(a) - h(b)| \leq L\|a - b\|_2 \quad \text{for all } a, b \in \mathbb{R}^m$$
2. Let c be β-Lipschitz smooth, i.e.
$$\|Jc(x) - Jc(y)\| \leq \beta\|x - y\|_2 \quad \text{for all } x, y \in \mathbb{R}^d$$
where $\|\cdot\|$ is the operator norm.

Under those assumptions and for $t \leq (L\beta)^{-1}$ the function F_t is an upper model for F. For a proof see, e.g. [15, section 3.3]. The convergence of the algorithm is monitored in terms of the *prox-gradient* mapping

$$\mathcal{G}_t\left(x^k\right) := \frac{1}{t}\left(x^k - x^{k+1}\right) \tag{4.6}$$

As shown in [15, section 4] a point x^k for which $\left\|\mathcal{G}_t\left(x^k\right)\right\|$ is small is a "nearly-stationary" point of F.

We now can state the prox-linear algorithm.

Algorithm 1: Prox-Linear Algorithm

Input: $x_0 \in \mathbb{R}^d$
Parameters: *Choose* $t \leq (L\beta)^{-1}$ *and precision* ε

while $\|\mathcal{G}_t(x_k)\| \geq \varepsilon$ **do**
  $x^{k+1} = \arg\min_x \left\{ g(x) + h\left(c\left(x^k\right) + Jc\left(x^k\right)\left(x - x^k\right)\right) + \frac{1}{2t}\left\|x - x^k\right\|_2^2 \right\}$
  $\mathcal{G}_t\left(x^k\right) = \frac{1}{t}\left(x^k - x^{k+1}\right)$
end

If this algorithm is equipped with a backtracking line search, we can prove linear convergence for the general case and even quadratic convergence under some additional conditions, as shown in [14]. Ochs et al. proposed a more traceable line search for the prox-linear algorithm [33].

[4] We use t to denote the step width here to stay consistent with the literature on the prox-linear method. The letter τ for the step width is more prominent in the PDHG literature

4.2.2 Prox-Linear Algorithm for Quadratic Regression

One special case is obtained when we apply the prox-linear algorithm to a nonlinear quadratic regression problem. Consider the problem introduced in Section 3.2.

$$\min_{\theta} F(\theta) = \frac{1}{2} \sum_{i=1}^{n} \big(\underbrace{y_i - f(x_i; \theta)}_{r_i} \big)^2 \tag{4.7}$$

where we denoted the differences between the model and the observations as residues r_i. By identifying $g \equiv 0$, $h(x) = \frac{1}{2}\|x\|^2$ and $c = \begin{bmatrix} r_1, & \dots & r_n \end{bmatrix}^T =: \mathbf{r}$ we can interpret Equation (4.7) as a composite optimization problem. Using the prox-linear algorithm we obtain the iteration

$$\theta^{k+1} = \arg\min_{\theta} \left\{ \frac{1}{2} \left\| \mathbf{r}\left(\theta^k\right) + J\mathbf{r}\left(\theta^k\right)\left(\theta - \theta^k\right) \right\|_2^2 + \frac{1}{2t} \left\| \theta - \theta^k \right\|_2^2 \right\} \tag{4.8}$$

To shorten the notation we use $\mathbf{r} = \mathbf{r}\left(\theta^k\right)$ and $J = J\mathbf{r}\left(\theta^k\right)$. The above minimization can be solved analytically, which yields

$$\theta^{k+1} = \theta^k - \left(J^T J + \frac{1}{t} I \right)^{-1} J^T \mathbf{r} \tag{4.9}$$

This is a special case of the method of "damped least suqares" introduced by Levenberg [27]. While Equation (4.9) is often referred to as the *Levenberg-Marquardt method*, Levenberg proposed the more general approach of component-wise damping which is obtained by replacing the identity matrix I by a positive definite diagonal matrix. Marquardt [29] did a more rigorous analysis on the algorithm and proposed to use the following diagonal damping term

$$\theta^{k+1} = \theta^k - \left(J^T J + \frac{1}{t} \operatorname{diag}\left(J^T J\right) \right)^{-1} J^T \mathbf{r} \tag{4.10}$$

where with a slight abuse of notation $\operatorname{diag}\left(J^T J\right)$ means the diagonal matrix containing the diagonal elements of $J^T J$. Although he did not analyze it, Levenberg reported the same damping term to be successful as well.

Note that the optimization problem associated to Equation (4.10) reads

$$\theta^{k+1} = \arg\min_{\theta} \left\{ \frac{1}{2} \left\| \mathbf{r}\left(\theta^k\right) + J\mathbf{r}\left(\theta^k\right)\left(\theta - \theta^k\right) \right\|_2^2 + \frac{1}{2} \left\| \theta - \theta^k \right\|_M^2 \right\} \tag{4.11}$$

with $M^{-1} = \frac{1}{t} \operatorname{diag}\left(J^T J\right)$. Therefore we can interpret Marquardts proposition as the application of the prox-linear approach with a weighted prox operator to a nonlinear quadratic regression problem.

Intuitively, Equation (4.10) is a component-wise damping which is proportional to an approximation of the curvature in this direction. To see this we consider the Hessian matrix of the square regression cost function $F(\theta) = \frac{1}{2}\|\mathbf{r}(\theta)\|_2^2$.

$$(HF)_{ij} = \sum_k \left(\frac{\partial r_k}{\partial \theta_i} \frac{\partial r_k}{\partial \theta_j} + r_k \frac{\partial^2 r_i}{\partial \theta_i \partial \theta_j} \right) \tag{4.12}$$

By dropping the second order term we see that we can approximate the Hessian matrix by $HF = J^T J$ where J is again the Jacobian matrix of the residues. Recall that

the diagonal entries of the Hessian matrix are the curvatures is the respective direction. Therefore, we see that in Equation (4.10) we are damping components with a high approximated curvature stronger than components with a lower curvature which helps to prevent overshooting.

Another beneficial aspect of using the derivatives is that the scaling of the components is introduced to the damping. The orders of magnitude can differ strongly between the components. In approaches like Equation (4.9), we still have to find one damping parameter that suits them all, which can lead to slow convergence for some components. This is avoided by the approach in Equation (4.10).

Remark. In the literature, we find diverse definitions of the Levenberg-Marquardt algorithm, most notably the versions defined by Equation (4.9) and Equation (4.10). While they both can be seen as special cases of a component-wise damped least-squares approach, they are still distinct methods, and we need to be careful which method is referred to when we talk about Levenberg-Marquardt.

4.2.3 Prox-Linear Algorithm with Weighted Prox Operator

To obtain more flexibility and to be able to account e.g. for different scales of the optimization variable, we will extend the standard prox-linear algorithm by using a weighted proximal mapping. This is motivated by the interpretation of the Levenberg-Marquardt approach discussed before. We will use a positive definite diagonal weighting matrix M^k which can be adjusted for each sub-problem. So the weighted prox-linear iteration is given as

$$x^{k+1} = \underset{x\in\mathbb{R}^d}{\arg\min} \left\{ g\left(x\right) + h\left(c\left(x^k\right) + Jc\left(x^k\right)\left(x - x^k\right)\right) + \frac{1}{2}\left\|x - x^k\right\|_{M^k}^2 \right\} \tag{4.13}$$

This modification of the standard algorithm proved to be working, and convergence improvements have been observed by assigning different step widths to the components. However, no rigorous convergence analysis has been done. Together with insights on the optimal choice of the weighting matrix M^k, this remains an exciting research perspective.

4.3 The Primal-Dual Hybrid Gradient Method

The primal-dual hybrid gradient algorithm (PDHG) is used to solve convex optimization problems of the form

Problem 4.1. Solve

$$\inf_{u\in\mathbb{R}^n} G\left(u\right) + F\left(Ku\right) \tag{4.14}$$

where $G : \mathbb{R}^n \to \overline{\mathbb{R}}$ and $F : \mathbb{R}^m \to \overline{\mathbb{R}}$ are proper, closed and convex functions and $K : \mathbb{R}^n \to \mathbb{R}^m$ is a linear operator.

The PDHG method was initially proposed by Pock, Bischof, Cremers and Chambolle for the minimization of the Mumford-Shah functional, see [36]. A more general discussion of the algorithm as well as a convergence analysis can be found in [7].

In the algorithm, we introduce a dual variable by replacing one of the functions F and G by its convex biconjugate. This transforms the problem into a saddle-point formulation that is solved by alternating steps on the primal and the dual

variable. The approach is useful in particular if the function F is non-smooth. While standard gradient-based methods are not applicable, the replacement with the biconjugate separates the linear operator from the function, which makes an effective solution possible.

In the following, we discuss the ideas behind the PDHG algorithm to build some intuition on how the method works. As the convergence proofs and the theory behind it are extensive, we only state necessary results and refer to the respective work for more information.

4.3.1 Basics from Convex Analysis

We start with a few definitions and results from convex analysis that are necessary to understand the idea behind the PDHG method. The proofs and a thorough discussion of the results can be found in the standard book by Rockafellar [40].

Definition 4.4 (Convex Conjugate)**.** Let $f : \mathbb{R}^n \cup \{\infty\}$ be any proper function. The the *convex conjugate* of f is defined as

$$f^*(p) = \sup_{u \in \mathbb{R}^n} \left(\langle p, u \rangle - f(u) \right)$$

where $\langle \cdot, \cdot \rangle$ denotes the standard Euclidean scalar product.

Besides being convex and closed, the convex conjugate has the following property, which provides the basis for the PDHG method.

Theorem 4.2 (Biconjugate of Convex Functions)**.** Let $f : \mathbb{R}^n \cup \{\infty\}$ be proper, convex and closed. Then the biconjugate is equal to f, i.e

$$f^{**}(u) := \sup_{q \in \mathbb{R}^n} \left(\langle q, u \rangle - f^*(q) \right) = f(u)$$

Under the assumptions stated in Problem 4.1 we can use this result to replace the function F by its biconjugate to re-write the problem as

$$\inf_{u \in \mathbb{R}^n} \sup_{q \in \mathbb{R}^n} G(u) + \langle q, Ku \rangle - F^*(q) \tag{4.15}$$

Under a mild assumption, we can interchange the infimum and the supremum, which allows the next statement.

Theorem 4.3 (Fenchel's Duality)**.** Let $G : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ and $F : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ be proper, closed and convex functions. Let $K : \mathbb{R}^n \to \mathbb{R}^m$ be a linear operator with the adjoint operator K^T. Let further exist a $x \in ri\left(dom\left(G\right)\right)$[5]such that $Ku \in ri\left(dom\left(F\right)\right)$. Then the solutions of the following problem formulations are equivalent

$$\inf_{u \in \mathbb{R}^n} G(u) + F(Ku) \qquad \text{Pimal Problem} \tag{4.16}$$

$$\inf_{u \in \mathbb{R}^n} \sup_{q \in \mathbb{R}^n} G(u) + \langle q, Ku \rangle - F^*(q) \qquad \text{Saddle} - \text{point Problem} \tag{4.17}$$

$$\sup_{q \in \mathbb{R}^n} \inf_{u \in \mathbb{R}^n} G(u) + \langle q, Ku \rangle - F^*(q) \qquad \text{Saddle} - \text{point Problem} \tag{4.18}$$

$$\sup_{q \in \mathbb{R}^n} -G^*\left(-K^T q\right) - F^*(q) \qquad \text{Dual Problem} \tag{4.19}$$

The dual formulation is obtained by replacing the function G by its biconjugate as well and some algebraic operations.

In the PDHG algorithm, we perform descend steps on the primal and ascend steps on the dual variable in an alternating fashion. We use the *proximal operator* to do so.

4.3.2 The PDHG Algorithm

We now can derive the primal dual hybrid gradient algorithm. We will solve the saddle-point formulation of the problem and therefore we will work with the following function

$$\mathcal{L}(u,p) := G(u) + \langle p, Ku\rangle - F^*(p)$$

with the primal variable u and the dual variable p. Fixing the current primal we first perform the following ascent step on the dual variable

$$\begin{aligned} p^{k+1} &= \operatorname{prox}_{-\sigma\mathcal{L}\left(u^k,\cdot\right)}\left(p^k\right) \\ &= \arg\min_p \left\{ \sigma F^*(p) - \sigma\langle p, Ku^k\rangle + \frac{1}{2}\left\|p - p^k\right\|_2^2 \right\} \\ &= \arg\min_p \left\{ \sigma F^*(p) + \frac{1}{2}\left\|p - p^k - \sigma K u^k\right\|_2^2 \right\} \\ &= \operatorname{prox}_{\sigma F^*}\left(p^k - \sigma K u^k\right) \end{aligned}$$

with the yet unspecified stepwidth σ. This is followed by a descent step on the primal variable that we can simplify in a similar way.

$$\begin{aligned} u^{k+1} &= \operatorname{prox}_{\tau\mathcal{L}\left(\cdot,p^{k+1}\right)}\left(u^k\right) \\ &= \arg\min_u \left\{ \tau G(u) + \tau\langle p^{k+1}, Ku\rangle + \frac{1}{2}\left\|u - u^k\right\|_2^2 \right\} \\ &= \arg\min_u \left\{ \tau G(u) + \frac{1}{2}\left\|u - u^k + \tau K^T p^{k+1}\right\| \right\} \\ &= \operatorname{prox}_{\tau G}\left(u^k - \tau K^T p^{k+1}\right) \end{aligned}$$

[5]The *relative interior* $ri(C)$ of a set C is a concept that is used, when the low-dimensional set C is embedded into a higher-dimensional space. Intuitively, it is the interior of the set C restricted to the smallest possible subspace.

By combining these two steps with an extrapolation step, we end up with the algorithm analyzed in [7].

Algorithm 2: Primal Dual Hybrid Gradient Algorithm

Input: $u^0 \in \mathbb{R}^n, p^0 \in \mathbb{R}^m$
Parameters: τ, σ *s.t.* $\sigma\tau < \frac{1}{\|K\|^2}$

Set $\bar{u}^0 = u^0$
for $k = 0, 1, 2, \dots$ **do**
  $p^{k+1} = \text{prox}_{\sigma F^*}\left(p^k + \sigma K \bar{u}^k\right)$
  $u^{k+1} = \text{prox}_{\tau G}\left(u^k - \tau K^T p^{k+1}\right)$
  $\bar{u}^{k+1} = 2u^{k+1} - u^k$
end

Here $\|K\|$ is the operator norm of K. Chambolle and Pock proved the convergence of the algorithm with rate $O(1/k)$ in [7].

4.4 Preconditiong for the PDHG Algorithm

To determine the step widths in Algorithm 2 we need to estimate $\|K\|$. If the operator has a complex structure, this might become a very demanding task. Also, we encounter operators that have a very large norm. As the resulting step widths are small, the convergence of the algorithm is very slow. To overcome these problems, Pock and Chambolle proposed a diagonal preconditioning approach as an extension to Algorithm 2 in [35]. Intuitively, the idea is to assign each component of the primal and the dual variable its customized step width, which is done by the weighted proximal mapping from Definition 4.2.

Algorithm 3: Primal Dual Hybrid Gradient Algorithm with Preconditioning

Input: $u^0 \in \mathbb{R}^n, p^0 \in \mathbb{R}^m$
Parameter: *Choose diagonal matrices* Λ, Σ *according to Theorem 4.4*

Set $\bar{u}^0 = u^0$
for $k = 0, 1, 2, \dots$ **do**
  $p^{k+1} = \text{prox}_{\Sigma F^*}\left(p^k + \Sigma K \bar{u}^k\right)$
  $u^{k+1} = \text{prox}_{\Lambda G}\left(u^k - \Lambda K^T p^{k+1}\right)$
  $\bar{u}^{k+1} = 2u^{k+1} - u^k$
end

Note that by the choice $\Lambda = \tau I$, $\Sigma = \sigma I$ with $\sigma\tau < \|K\|$ we recover the standard PDHG algorithm. Pock and Chambolle proved convergence of Algorithm 3 for the following choice of preconditioning matrices

Theorem 4.4. Let $\alpha \in [0, 2]$. The choice $\Lambda = \text{diag}(\tau_1 \dots, \tau_n)$ and $\Sigma = \text{diag}(\sigma_1, \dots, \sigma_m)$ with

$$\tau_j = \frac{1}{\sum_{i=1}^m |K_{ij}|^{2-\alpha}}, \quad \sigma_i = \frac{1}{\sum_{j=1}^n |K_{ij}|^{\alpha}} \tag{4.20}$$

ensures that the sequence $\left(u^k, p^k\right)$ generated by Algorithm 3 converges weakly to an optimal solution of the saddle-point formulation of Problem 4.1

5 Implementation

In this chapter, we discuss the implementation of the prox-linear algorithm to solve the SLAM optimization problem. While we use the PDHG method to solve the resulting sub-problems in the general case, we also discuss an analytic solution of the sub-problems for a special case to be able to evaluate the performance as well as advantages and disadvantages of the PDHG approach. Starting from Problem 3.1, we address the different aspects of the approaches.

5.1 Vectorization of the depth map

For the implementation we will use a vectorized form of the depth map. This will simplify the notation when we need to calculate the Jacobian matrix of the image intensities w.r.t the optimization variable. The vectorization is obtained by stacking the columns, i.e. for $\mathbf{h} = [\mathbf{h}_1, \dots, \mathbf{h}_n]$ we write the vectorized version as

$$\mathbf{h}^{vec} = \begin{bmatrix} \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_n \end{bmatrix}$$

Analogously, we can vectorize the image intensity as well as the image gradients. The transformation between pixel coordinates and the index in the vectorized quantity is stated in the next proposition.

Proposition 5.1. Let $\mathbf{x} = (x, y) \in \Omega$ be a point in the image domain which we can assign a value in the matrix $\mathbf{h} \in \mathbb{R}^{m \times n}$ via the mapping in Definition 2.1. Then the same value can be found in the vectorization of the matrix $\mathbf{h}^{vec}$ at the index

$$i = y + m\,(x - 1)$$

Vice versa the i-th entry in the vectorization can be found in the matrix at the integer point

$$\begin{aligned} x &= \left\lfloor \frac{i-1}{m} \right\rfloor + 1 \\ y &= ((i-1) \mod m) + 1 \end{aligned}$$

Where $\lfloor \cdot \rfloor$ denotes the floor function and $\mod$ the modulo operation.

Using the vectorized form of the depth map, we can write the optimization variable compactly as a vector u.

Definition 5.1. We write the optimization variable for the implementation as

$$u = \begin{bmatrix} \mathbf{w} \\ \mathbf{T} \\ \mathbf{h}^{vec} \end{bmatrix}$$

Note that $\mathbf{h}$ and $\mathbf{h}^{vec}$ are merely different representations of the same quantity, and we will still use the matrix representation whenever it suits us.

The vectorization of the depth map also allows us to write the forward difference operators and, consequently, the complete discrete gradient operator as matrices.

We define $D_x, D_y \in \mathbb{R}^{mn \times mn}$ such that $D_x \mathbf{h}^{vec}$ yields a vector that contains the forward differences in x-direction for all pixels in a stacked ordering (i.e. the same ordering as in $\mathbf{h}^{vec}$). Likewise, $D_y \mathbf{h}^{vec}$ contains the forward differences in y-direction. The written-out difference matrices can be found in Appendix B.1.

It will proof useful to combine both finite difference matrices into one discrete gradient operator that acts on the complete optimization variable. Hence we define the matrix

$$\bar{D} = \begin{bmatrix} \mathbf{0} & D_x \\ \mathbf{0} & D_y \end{bmatrix} \tag{5.1}$$

where we added $6 \cdot |\mathcal{I}|$ zero columns at the beginning to account for the pose components of the optimization variable.

5.2 Application of the Prox-Linear Algorithm

First we show that we can apply the prox-linear algorithm to Problem 3.1. To see this and to prepare for the linearization we reformulate the dataterm slightly. We start with the simpler case of only two images, i.e. with Equation (3.5). By stacking the intensity differences for all the valid points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\} \in \mathcal{V}_{I_1}$ we obtain a vector

$$c(u) = \begin{bmatrix} \mathcal{J}I_1\Big(\omega_{\mathbf{x}_1}\big(R(\mathrm{w}), T, \mathbf{h}\big)\Big) - I_0(\mathbf{x}_1) \\ \mathcal{J}I_1\Big(\omega_{\mathbf{x}_2}\big(R(\mathrm{w}), T, \mathbf{h}\big)\Big) - I_0(\mathbf{x}_2) \\ \vdots \\ \mathcal{J}I_1\Big(\omega_{\mathbf{x}_n}\big(R(\mathrm{w}), T, \mathbf{h}\big)\Big) - I_0(\mathbf{x}_n) \end{bmatrix} \tag{5.2}$$

By introducing the function $h(\mathbf{y}) = \sum_j \ell(y_j)$ for $\mathbf{y} = [y_1, y_2, \ldots, y_n]^T$ we can write the data term as

$$E_{data}(\mathrm{w}, T, \mathbf{h}) = h(c(u)) \tag{5.3}$$

The dataterm for multiple images from Equation (3.6) can be rewritten analogously. In this case we define a vector similar to the one in Equation (5.2) for each available image $i \in \mathcal{I}$ each.

$$c_i(u) = \begin{bmatrix} \mathcal{J}I_i\Big(\omega_{\mathbf{x}_1}\big(R(\mathrm{w_i}), T_i, \mathbf{h}\big)\Big) - I_0(\mathbf{x}_1) \\ \mathcal{J}I_i\Big(\omega_{\mathbf{x}_2}\big(R(\mathrm{w_i}), T_i, \mathbf{h}\big)\Big) - I_0(\mathbf{x}_2) \\ \vdots \\ \mathcal{J}I_i\Big(\omega_{\mathbf{x}_{n_i}}\big(R(\mathrm{w_i}), T_i, \mathbf{h}\big)\Big) - I_0(\mathbf{x}_{n_i}) \end{bmatrix} \tag{5.4}$$

Note that the c_i will have different lengths as the set of valid points $\mathcal{V}_{I_i}$ is different for each image. By stacking the vectors c_i we write the dataterm for multiple images as

$$E_{data}(\mathbf{w}, \mathbf{T}, \mathbf{h}) = h(c(u)) = h\left(\begin{bmatrix} c_1(u) \\ \vdots \\ c_{|\mathcal{I}|}(u) \end{bmatrix}\right) \tag{5.5}$$

So we can write the SLAM problem as

$$\min_u \{h(c(u)) + \lambda_{reg} E_{reg}(u)\} \tag{5.6}$$

It is easy to show that all the loss functions introduced in Section 3.2 are closed and convex. Therefore h is closed and convex as it is the sum of closed and convex functions. Likewise, the regularization term can be shown to be closed and convex as it is a composition of a linear map, a convex norm, and a non-negative weighted sum. Note that the (positive) adaptive weights do not depend on the optimization variable and therefore have no influence on the convexity.

The last prerequisite for the composite optimization problem from Definition 4.3 is that c is continuously differentiable. The function is a composition of the smooth warping ω and the image interpolation. Therefore the differentiability of c depends on the interpolation method. If we choose the bicubic convolution approach, we obtain a C^1 function c and hence can apply the prox-linear approach. However, for performance reasons, often the bilinear approach is preferred, which is not differentiable. In this case, a continuous approximation of the gradient is obtained by bilinear interpolation of the image gradients at the pixel centers which are computed via finite differences. To avoid systematic errors, it is important to use an unbiased finite difference operator like central differences for that purpose.

5.2.1 Linearization of c

For the prox-linear algorithm we need to linearize c at the current iterate u^k. Considering one of the c_i we can write the linearization as

$$\begin{aligned} c_i^{lin}(u) &= c_i\left(u^k\right) + Jc_i\left(u^k\right)\left(u - u^k\right) \\ &= J_i^k u - b_i^k \end{aligned} \tag{5.7}$$

where we shortened the notation for the Jacobian $Jc_i\left(u^k\right)$ to J_i^k and introduced the vector

$$\begin{aligned} b_i^k &= J_i^k u^k - c_i\left(u^k\right) \\ &= J_i^k u^k - \begin{bmatrix} \mathcal{J}I_i\Big(\omega_{\mathbf{x}_1}\left(R\left(\mathrm{w}_i^k\right), T_i^k, \mathbf{h}^k\right)\Big) - I_0\left(\mathbf{x}_1\right) \\ \mathcal{J}I_i\Big(\omega_{\mathbf{x}_2}\left(R\left(\mathrm{w}_i^k\right), T_i^k, \mathbf{h}^k\right)\Big) - I_0\left(\mathbf{x}_2\right) \\ \vdots \\ \mathcal{J}I_i\Big(\omega_{\mathbf{x}_{n_i}}\left(R\left(\mathrm{w}_i^k\right), T_i^k, \mathbf{h}^k\right)\Big) - I_0\left(\mathbf{x}_{n_i}\right) \end{bmatrix} \end{aligned} \tag{5.8}$$

By combining the linearizations for all available images we now can write down the linearized form of the dataterm which we use for the prox-linear algorithm

$$E_{data}^{lin}(u) = h\left(\begin{bmatrix} J_1^k \\ \vdots \\ J_{|\mathcal{I}|}^k \end{bmatrix} u - \begin{bmatrix} b_1^k \\ \vdots \\ b_{|\mathcal{I}|}^k \end{bmatrix}\right) = h\left(J^k u - b^k\right) \tag{5.9}$$

So the prox-linear subproblem in every step reads

$$u^{k+1} = \arg\min_u \left\{ h\left(J^k u - b^k\right) + \lambda_{reg} E_{reg}(u) + \frac{1}{2}\left\|u - u^k\right\|_{M^k}^2 \right\} \tag{5.10}$$

where we used the weighted prox-linear approach with a weighting matrix M^k that can change for each iteration. The reason for this as well as the actual choice will be discussed in Section 5.2.2.

It remains to address the computation of the matrices J_i^k. For an arbitrary point $\mathbf{x}_j$ the corresponding row in J_i^k is the transposed of the following gradient

$$\begin{bmatrix} \nabla_{\mathbf{w}} \\ \nabla_{\mathbf{T}} \\ \nabla_{\mathbf{h}^{vec}} \end{bmatrix} (\mathcal{J}I_i) \left(\omega_{\mathbf{x}_j} \left(R\left(\mathrm{w_i}\right), T_i, \mathbf{h} \right) \right) \tag{5.11}$$

In accordance with Section 2.4 we denote the warping of point $\mathbf{x}_j$ to image i by

$${}_i\mathbf{x}_j = \omega_{\mathbf{x}_j} \left(R\left(\mathrm{w_i}\right), T_i, \mathbf{h} \right) \tag{5.12}$$

Recall that $\mathbf{x}_j$ is actually short for ${}_0\mathbf{x}_j$, i.e. a point in the reference image.

It is immediate that we will need to apply the chain rule to calculate the gradient in Equation (5.11). By using the calculus rules in Appendix A we can write down the first component of the gradient as

$$\nabla_{\mathbf{w}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) = \left(J_{\mathbf{w}} \omega_{\mathbf{x}_j} \right)^T \nabla_{\mathbf{x}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) = \left(J_R \omega_{\mathbf{x}_j} J_{\mathbf{w}} R \right)^T \nabla_{\mathbf{x}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) \tag{5.13}$$

where we omitted the evaluation points for the Jacobian matrices to improve readability. Recall that an index for the Jacobian matrix means the Jacobian matrix w.r.t only this vector and note that $\nabla \left(\mathcal{J}I_i\right)$ in the sense of the nabla operator is only defined if we use the bicubic interpolation. In the bilinear case the notation means the bilinear interpolation of the image gradients at the pixel centers. The formulas for both cases are given in Appendix B.2. Analogously we can write

$$\nabla_{\mathbf{T}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) = \left(J_{\mathbf{T}} \omega_{\mathbf{x}_j} \right)^T \nabla_{\mathbf{x}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) \tag{5.14}$$

$$\nabla_{\mathbf{h}^{vec}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) = \left(J_{\mathbf{h}^{vec}} \omega_{\mathbf{x}_j} \right)^T \nabla_{\mathbf{x}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right) \tag{5.15}$$

Combining the previous results the j-th row of J_i^k reads

$$\nabla_{\mathbf{x}} \left(\mathcal{J}I_i\right) \left({}_i\mathbf{x}_j\right)^T \begin{bmatrix} J_R \omega_{\mathbf{x}_j} J_{\mathbf{w}} R & J_{\mathbf{T}} \omega_{\mathbf{x}_j} & J_{\mathbf{h}^{vec}} \omega_{\mathbf{x}_j} \end{bmatrix} \tag{5.16}$$

We note that the Jacobian matrix $J_R\omega$ is not defined in the current form of the warping formula, as R is a matrix, and our notation only allows Jacobian matrices w.r.t vectors. Therefore, we need to reformulate the formula to use a vectorized form of the rotation matrix. We use the following standard result from linear algebra.

Lemma 5.1. Let $A \in \mathbb{R}^{k \times l}, B \in \mathbb{R}^{m \times n}$ and $C \in \mathbb{R}^{k \times n}$ be matrices. Let $X \in \mathbb{R}^{l \times m}$ be another matrix such that $AXB = C$ holds. Then the following equivalency holds

$$AXB = C \Leftrightarrow \left(B^T \otimes A \right) \mathrm{vec}\left(X\right) = \mathrm{vec}\left(C\right)$$

where the operator vec means the stacking operation of the columns of the matrix, and $\otimes$ is the Kronecker product.

We start from Equation (2.24) which we adjust for the case that we warp a general point $\mathbf{x}$ from the reference frame to frame i.

$${}_ih_i\tilde{\mathbf{x}} = hKRK^{-1}\tilde{\mathbf{x}} + KT \tag{5.17}$$

Since $KRK^{-1}\tilde{\mathbf{x}}$ yields a vector we can use Lemma 5.1 to write

$$KRK^{-1}\tilde{\mathbf{x}} = \left(\left(K^{-1}\tilde{\mathbf{x}} \right)^T \otimes K \right) R^{vec} \tag{5.18}$$

where we use the same notation for the vectorization of R as for the depth map. By introducing $\Psi = \left(K^{-1}\tilde{\mathbf{x}}\right)^T \otimes K$ we can rewrite Equation (5.17) as

$$ {}_ih \begin{bmatrix} {}_i\mathbf{x} \\ 1 \end{bmatrix} = h\Psi R^{vec} + KT \tag{5.19} $$

from which we obtain the reformulated warping equation similarly to Section 2.4 as

$$ {}_i\mathbf{x} = \omega_{\mathbf{x}}\left(R, T, \mathbf{h}\right) = \frac{1}{h\Psi_3 R^{vec} + K_3 T} \begin{bmatrix} h\Psi_1 R^{vec} + K_1 T \\ h\Psi_2 R^{vec} + K_2 T \end{bmatrix} \tag{5.20} $$

where Ψ_i means the i-th row of the matrix.

For Equation (5.20) it is now possible to calculate the Jacobian matrix $J_{R^{vec}}\omega$. By vectorizing the result of Rodrigues Formula given in Theorem 2.1, we can also compute $J_{\mathbf{w}}R^{vec}$, which now finally allows the computation of the complete Jacobian matrices J_i.

5.2.2 Choice of the Stepwidths

Since we have not established a rigorous theory for the choice of the diagonal step-width matrices M^k, we use the following heuristic in the prox-linear approach.

$$ \left(M^k\right)^{-1} = \frac{1}{\zeta_{step}^k} M_0^{-1} + \min\left\{\operatorname{diag}\left(J^{k^T} J^k\right), M_{min}^{-1}\right\} \tag{5.21} $$

where for $0 < \zeta_{step} < 1$ we denote an actual exponentiation by ζ_{step}^k (in contrast to the iteration counts).

The first term corresponds to an exponential step width reduction starting from an initial weighting matrix M_0. Such a reduction rule is quite common in optimization and is used to avoid overshooting near the optimum. Moreover, it is sensible to reduce the step width in combination with the iteratively adjusted blurring introduced in the next section, as this affects the smoothness of the cost function.

The second term is inspired by the Levenberg-Marquardt approach and can be seen as an adaptive damping of components where the cost function has a high curvature in the respective direction. The reasoning is analogous to Section 4.2.2. The minimization in the second term, which is meant component-wise, is used to limit the influence of the damping term. M_{min} is a component-wise minimum step width, and its inverse limits the diagonal term.

We decompose the step width parameters such that we use the same parameter for each component of translation, rotation and depth, respectively. For example, for the initial step width we use

$$ M_0 = \begin{bmatrix} M_{0,\mathrm{w}} \\ M_{0,T} \\ M_{0,\mathbf{h}} \end{bmatrix} \tag{5.22} $$

where each $M_{0,*}$ is a vector with identical values and the suitable dimension to fit the respective part of the optimization variable.

Figure 15: *Effect of the Gaussian blurring. On the left the original image, in the middle the blurred image with standard deviation $\sigma = 1$ and on the right with $\sigma = 2$. Source image: Built-in examples in MATLAB, cameraman.tif*

5.2.3 Blurring to Increase the Stepwidths

The feasible step width for the prox-linear algorithm depends on the Lipschitz constant of the function h and the Lipschitz smoothness constant of the function c. It is well known that for twice continuously differentiable functions, the L-smoothness constant is coupled to the norm of the second derivative. A function with a lower curvature has a smaller Lipschitz smoothness constant. Hence, if we reduce the curvature of the function c in the prox-linear algorithm, this means that we can allow bigger step widths.

In the SLAM problem, we can smoothen the function c by applying a Gaussian filter to the input images. The effect on an image is shown in Figure 15. A summary of Gaussian filtering is given in Appendix B.4. With increasing standard deviation σ_{Gauss}, the filter increasingly equalizes the brightness values of the pixels, which are used as sampling points for the interpolation. Consequently, the curvature of the interpolation is reduced, which is conveyed to the function c.

However, the blurring of the images affects the sharpness of details. Therefore the estimated depth map might become more fuzzy with increasing blurring. To prevent this effect, we reduce the blurring of the images by a constant factor ζ_{blurr} with increasing number of iterations. As a result, we can increase the step width in the first iterations to increase convergence and are still able to obtain sharp results. Note that the heuristic approach for the step widths defined in Section 5.2.2 reduces the step widths, so the increase of the Lipschitz smoothness constant due to the reduction of the blurring does not cause any problems for a proper choice of the parameters.

In the tests, it turned out to be best to reduce the blurring only every r-th step and to keep it constant in between. Consequently, starting from an initial standard deviation $\sigma_{Gauss,0}$, the standard deviation for the blurring in step k reads

$$\sigma_{Gauss}^{k} = \zeta_{blurr}^{\left\lfloor \frac{k}{r} \right\rfloor} \sigma_{Gauss,0} \tag{5.23}$$

where we mean an actual exponentiation for the factor ζ_{blurr} while in σ_{Gauss}^{k} we denote the iteration counter.

5.2.4 Initial Value

Since the cost function for the SLAM problem is highly non-convex, we need to have a good initial guess to be able to estimate the pose and the depth map accurately. The estimation of good initial values is a problem of its own, which is beyond the scope of this thesis. One possible approach is to perform a repeated exhaustive epipolar line search[6]for each pixel to obtain an initialization. This approach is described e.g. in [18]. The initialization used for the experiments in this thesis is discussed in Section 6.1.3.

5.2.5 Pseudo Code for the Application of the Prox-Linear Algorithm

The results of this section are summarized in Algorithm 4.

5.3 Application of the PDHG Algorithm to the sub-problems

We now apply the PDHG algorithm to the sub-problems defined by Equation (5.10). We will discuss different loss functions for the data term and the isotropic version of both the TV regularization as well as the Huber regularization term with adaptive weights. To prepare for the PDHG formalism we need to re-write the regularization terms slightly. Using the discrete gradient operator $\bar{D}$ defined in Equation (5.1) we can write

$$E_{reg}(u) = \sum_{i=1}^{mn} \Gamma_i \left\| \begin{bmatrix} (\bar{D}u)_i \\ (\bar{D}u)_{i+mn} \end{bmatrix} \right\|_{\blacklozenge} \tag{5.24}$$

where Γ is the vector containing the stacked adaptive weights for the respective pixels and $\blacklozenge$ is a placeholder for either the 2-norm or the Huber norm. We now introduce the function

$$\mathcal{R}(\mathbf{x}) = \sum_{i=1}^{mn} \Gamma_i \left\| x_{[i,i+mn]} \right\|_{\blacklozenge} \tag{5.25}$$

where we use the indexing notation

$$x_{[i,i+mn]} = \begin{bmatrix} x_i \\ x_{i+mn} \end{bmatrix} \tag{5.26}$$

This allows us to write the regularization term as $E_{reg}(u) = \mathcal{R}\left(\bar{D}u\right)$.

As we have 3 terms in the current formulation of the optimization problem we need to take a few reformulation steps to bring the problem to the standard saddle point formulation in Theorem 4.3. We repeat Equation (5.10) here for convenience where we already used the new notation for the regularization term.

$$u^{k+1} = \arg\min_u \left\{ h\left(J^k u - b^k\right) + \lambda_{reg} \mathcal{R}\left(\bar{D}u\right) + \frac{1}{2} \left\| u - u^k \right\|_{M^k}^2 \right\} \tag{5.27}$$

[6]In short, epipolar geometry describes geometric constraints between a 3D world point and its image points in different camera frames. One of those constraints is, that for a fixed camera pose the warping locations of a point from the reference camera to a second camera are restrained to a straight line for arbitrary depth values.

Algorithm 4: Prox-Linear Algorithm for the SLAM Problem

Input: I_i for $i \in \mathcal{I}, u^0 \in \mathbb{R}^n$

Parameters:

- *Interpolation method*
- *Loss function, regularization method and Huber parameters if applicable*
- *Regularization weight λ_{reg}, adaptive regularization weight parameter α, β*
- *Blurring reduction parameters $\sigma_{Gauss,0}, \zeta_{blurr}$ and r*
- *Stepwidth parameters M_0, M_{max} and ζ_{step}*

Preprocessing:

- *Apply initial blurring to images*
- *Calculate adaptive regularization weights based on the original image*

for $k = 0, 1, 2, \ldots$ **do**

1. Check if blurring is reduced
2. Warp all points from reference image to secondary images
3. Check point validity
4. Interpolate secondary images at valid points
5. Compute linearization quantities J^k and b^k
6. Compute current step width matrix M^k
7. Solve sub-problem and update optimization variable

$$u^{k+1} = \arg\min_u \left\{ h\left(J^k u - b^k\right) + \lambda_{reg} E_{reg}(u) + \frac{1}{2} \left\| u - u^k \right\|_{M^k}^2 \right\}$$

end

We replace the two terms $F_1\left(J^k u\right) = h\left(J^k u - b^k\right)$ and $F_2\left(u\right) = \lambda_{reg}\mathcal{R}\left(\bar{D}u\right)$ by their biconjugates. In doing so we introduce two dual variables. We can re-write Equation (5.27) as

$$\min_u \max_{p_1} \max_{p_2} \langle p_1, J^k u\rangle - F_1^*\left(p_1\right) + \langle p_2, \bar{D}u\rangle - F_2^*\left(p_2\right) + \frac{1}{2}\left\|u - u^k\right\|_{M^k}^2 \tag{5.28}$$

By stacking the two dual variables into a new variable p and introducing $F^*\left(p\right) = F_1^*\left(p_1\right) + F_2^*\left(p_2\right)$ we obtain the standard form for the PDHG algorithm.

$$\min_u \max_p \underbrace{\frac{1}{2}\left\|u - u^k\right\|_{M^k}^2}_{G(u)} + \left\langle \underbrace{\begin{bmatrix} p_1 \\ p_2 \end{bmatrix}}_{:=p}, \underbrace{\begin{bmatrix} J^k \\ \bar{D} \end{bmatrix}}_{:=K} u \right\rangle - F^*\left(p\right) \tag{5.29}$$

To this formulation we can apply the standard or the preconditioned PDHG algorithm. Note that in the preconditioned case we can split the preconditioning matrix Σ into two parts for each of the dual variables. Closed form solutions for the prox operators can be found in Appendix B.3.

5.4 Analytic Solution of the Sub-Problems - Special Case

To be able to compare the performance of the general approach discussed before, we consider a special case for which we can derive an analytic solution for the sub-problems. We use the quadratic loss for the data term and the isotropic Huber regularization. This special problem reads

$$\min_u \left\{ \frac{1}{2}\left\|c\left(u\right)\right\|_2^2 + \lambda_{reg}\mathcal{R}\left(\bar{D}u\right) \right\} \tag{5.30}$$

To apply the prox-linear formalism we define the functions

$$h\left(\begin{bmatrix} \mathbf{x} \\ y \end{bmatrix}\right) = \frac{1}{2}\left\|x\right\|_2^2 + y; \quad \bar{c}\left(u\right) = \begin{bmatrix} c\left(u\right) \\ \lambda_{reg}\mathcal{R}\left(\bar{D}u\right) \end{bmatrix} \tag{5.31}$$

which allows us to write the SLAM problem as

$$\min_u h\left(\bar{c}\left(u\right)\right) \tag{5.32}$$

It is easy to see that h and $\bar{c}$ fulfill the prerequisites for the of the composite optimization problem in Definition 4.3 with the limitations due to the interpolation as discussed in Section 5.2. To apply the prox-linear algorithm we need to linearize $\bar{c}$ at the current iterate u^k.

$$\bar{c}^{lin} = \begin{bmatrix} c\left(u^k\right) \\ \lambda_{reg}\mathcal{R}\left(\bar{D}u^k\right) \end{bmatrix} + \begin{bmatrix} Jc\left(u^k\right) \\ \lambda_{reg}\nabla\left(\mathcal{R}\left(\bar{D}u^k\right)\right)^T \end{bmatrix}\left(u - u^k\right) \tag{5.33}$$

Where we need to apply the chain rule to compute the gradient of the composition $\mathcal{R}\left(\bar{D}u^k\right)$, i.e.

$$\nabla\left(\mathcal{R}\left(\bar{D}u^k\right)\right) = \bar{D}^T\nabla\mathcal{R}\left(\bar{D}u^k\right) \tag{5.34}$$

The formula for the gradient of $\mathcal{R}\left(\mathbf{x}\right)$ is given in Appendix B.5.

Using the linearization we obtain the following iteration of sub-problems by applying the weighted prox-linear algorithm.

$$u^{k+1} = \arg\min_u \left\{ \frac{1}{2} \left\| J^k u - b^k \right\|_2^2 + \lambda_{reg} \nabla \mathcal{R}^T \left(\bar{D} u^k \right) \bar{D} u + \frac{1}{2} \left\| u - u^k \right\|_{M^k}^2 \right\} \tag{5.35}$$

where we used the same notation for J^k and b^k as before and already omitted the constant terms. As the sub-problems are quadratic forms we are able to compute an analytic solution by considering the optimality condition.

$$\begin{aligned} & J^{k^T} \left(J^k u - b^k \right) + \lambda_{reg} \bar{D}^T \nabla \mathcal{R} \left(\bar{D} u^k \right) + \left(M^k \right)^{-1} \left(u - u^k \right) = 0 \\ \Leftrightarrow \quad & u^{k+1} = \underbrace{\left(J^{k^T} J^k + \left(M^k \right)^{-1} \right)^{-1}}_{\Phi} \left(J^{k^T} b^k + \left(M^k \right)^{-1} u^k - \lambda_{reg} \bar{D}^T \nabla \mathcal{R} \left(\bar{D} u^k \right) \right) \end{aligned} \tag{5.36}$$

Due to the dimensionality of the problem, it is not possible to solve the linear system with standard algorithms. However, the matrix Φ has a special structure that allows us to solve the system using the Schur complement, a well-known approach e.g. in bundle adjustment algorithms. For completeness, the solution of the linear system using the Schur complement is given in Appendix B.6. Equation (5.36) replaces step 7 in Algorithm 4.

Remark. It is worth noting, that this approach can be seen as a mixture of a Levenberg-Marquardt-like approach on the data term and the gradient descent step on the regularization in the following way. First, we consider only the data term and do a Levenberg-Marquardt step. The result reads

$$u^{k+1} = \arg\min_u \left\{ \frac{1}{2} \left\| J^k u - b^k \right\|_2^2 + \frac{1}{2} \left\| u - u^k \right\|_{M_{LM}}^2 \right\} \tag{5.37}$$

where a general weighting matrix M_{LM} was used.

For the regularization term we use the fact that for a function $f : \mathbb{R}^n \to \mathbb{R}$ which is L-Lipschitz smooth, the following inequality holds

$$f(y) \leq f(x) + \nabla f(x)^T (y - x) + \frac{L}{2} \|y - x\|_2^2 \tag{5.38}$$

This inequality can be used to derive a L-dependent step width rule for a gradient descent approach. It also provides an upper model at a current iterate, which is what we will be using.

The regularization term can be shown to be L-Lipschitz smooth. We do not analyze the constant here. We now can create an upper model via

$$\lambda_{reg} \mathcal{R} \left(\bar{D} u \right) \leq \lambda_{reg} \mathcal{R} \left(\bar{D} u^k \right) + \nabla \left(\mathcal{R} \left(\bar{D} u^k \right) \right) \left(u - u^k \right) + \frac{L}{2} \left\| u - u^k \right\|^2 \tag{5.39}$$

By combining the Levenberg Marquardt approach for the pure data term with the minimization of the upper model defined in Equation (5.39) we obtain the following optimization problem

$$u^{k+1} = \arg\min_u \left\{ \frac{1}{2} \left\| J^k u - b^k \right\|_2^2 + \lambda_{reg} \nabla \mathcal{R}^T \left(\bar{D} u^k \right) \bar{D} u + \frac{1}{2} \left\| u - u^k \right\|_M^2 \right\} \tag{5.40}$$

where we defined $M^{-1} = M_{LM}^{-1} + LI$ with the identity matrix I, used Equation (5.34), and dropped constant terms. This is equivalent to Equation (5.35).

Regularization Term	
λ_{reg}	0.2
h_{reg}	1

Adaptive Regularization weight	
α_{reg}	$1 \cdot 10^{-5}$
β_{reg}	4

Image Blurring	
$\sigma_{Gauss,0}$	6
ζ_{blurr}	0.65
r	3

Prox-Linear Algorithm	
Number of Linearizations	30
ζ_{step}	0.9
$M_{0,\mathrm{w}}$	$2 \cdot 10^{-4}$
$M_{0,T}$	$1 \cdot 10^{-5}$
$M_{0,\mathbf{h}}$	50
$M_{min,\mathrm{w}}$	$2 \cdot 10^{-8}$
$M_{min,T}$	$5 \cdot 10^{-7}$
$M_{min,\mathbf{h}}$	$4 \cdot 10^{-1}$

PDHG Algorithm	
Number of Iterations	300
α (Preconditioning)	0.65

Table 1: *Standard Parameters for the Estimation*

6 Results

To test our approach, we use two synthetically created datasets. The New Tsukuba stereo dataset is an indoor stereo data set provided by the Computer Vision Laboratory of the University of Tsukuba [30, 34]. Since we are following a monocular approach we are only using the left image of this data. The images contain a lot of fine detail and therefore we will do most of the evaluation of the approach using this data set.

The second data set is an outdoor scene in an urban environment and was created using CARLA, an open source simulator for autonomous driving research [13]. The images contain a lot less detail, and we will use the CARLA sequence merely to show qualitatively that we did not overfit on the other dataset.

Both data sets contain the ground truth for the depth and the camera poses such that a proper evaluation of the performance of the algorithm is possible. Also, no distortion is present in the images, so we do not need any pre-processing steps. The implementation of the algorithm is done in MATLAB without the use of the GPU or any parallelization.

In the first section of this chapter we address several aspects that concern all the experiments that we conducted. This includes a set of standard parameters, the methodology used to evaluate the results an the generation of initial values. Afterwards, we discuss different experiments that evaluate different aspects of the approach.

6.1 General Considerations

6.1.1 Standard Parameters

We introduce a set of standard parameters for the cost function that we will use in multiple experiments. Table 1 lists the values. Recall that for the step width parameters of the prox-linear algorithm we use one single value for all components of the rotation, translation and depth, respectively, as described in Section 5.2.2.

By default we use the absolute loss and bilinear interpolation for the data term and the isotropic Huber approach for the regularization. We have tested both the bilinear as well as the bicubic approach for the interpolation and did not observe any significant difference in performance or quality of the result.

Any parameters not listed here are specific to the experiments and will be given in the respective sections. The same is true, if parts of the standard parameters needs to be changed for an experiment.

6.1.2 Error Measures for the Results

The potential scale drift presents a difficulty for the evaluation of the depth map and the translation vector. Since the monocular approach is unable to prevent a scale drift, we attempt to subtract out any scale drift present, in order to evaluate only what the algorithm is actually capable of doing.

Recall the definition of the scale drift in Section 3.5. We see, that if we are given the scaling factor ρ, we can use it to scale the translation and the depth map to the correct scale and evaluate the pure estimation errors. The problem is, that we cannot compute ρ, but rather need to estimate it.

For the moment, assume that we are given an estimate $\hat{\rho}$ for the scaling factor that we can use to correct the scale. To evaluate the error of the depth map we compute a pixel-wise signed relative error.

$$\delta_{\mathbf{h_i}} = \frac{\mathbf{h}_{true,i} - \hat{\rho}\mathbf{h}_i}{|\mathbf{h}_{true,i}|} \tag{6.1}$$

This error measure allows us to evaluate for each pixel if the result is too close or too far away, which can be illustrated in a heatmap.

For the translation we use the norm of the difference in the translation vectors to compute a standard relative error.

$$\delta_T = \frac{\|T_{true} - \hat{\rho}T\|_2}{\|T_{true}\|_2} \tag{6.2}$$

Note that we are using relative errors for both quantities to ensure comparability between different scenes and image pairs.

As it turns out, the error for the translation is highly sensitive to $\hat{\rho}$ and even small changes can have a big impact. Therefore, we need to treat the results for the pose with caution. For comparison we will sometimes plot the relative error of the uncorrected translation vector as well.

We now address the estimation of the scaling factor. Recall, that we need the scaling factor only for the computation of the errors, and that it does not affect the algorithm itself. Since the correct scaling factor should make the errors as small as possible compared to a wrong scaling factor for which the errors should be bigger, we compute $\hat{\rho}$ by minimizing the relative errors as a function of the scaling factor. To allow for fast computation we make the following approach.

$$\hat{\rho} = \arg\min_{\rho} \left\{ \delta_T^2 + \sum_i \delta_{\mathbf{h_i}}^2 \right\} \tag{6.3}$$

Instead of the original errors, we minimize the squared errors, which allows the analytic solution of the minimization. The drawback of using the squared errors is

the non-robustness towards outliers. Since we can have huge outliers, in particular in the depth map, we need to address this problem to ensure a reliable estimation. We do so by excluding all pixels from the sum, for which the unscaled and unsigned relative error is greater than 0.5. This is an heuristic choice that proved functional.

We will be evaluating the scaling factor over the iterations. Due to the high noise in the beginning, the estimation will not be very reliable for the first few steps, even with the outlier detection described before. Sill, we can assess the long term behavior of the scale and we observe the convergence of the estimator.

Note that the scale factor estimator is only a tool for the evaluation of the results and cannot be used in a real system, since it requires the ground truth. In Section 6.4 we analyze the performance of the estimator in the presence of a significant scale factor error.

Remark. There are other possibilities to estimate the scale factor ρ. We did try two alternatives, namely scaling the translation vector T to the length of the true translation and scaling the median of the depth values to the median of the true depth values. The first approach reduced the relative error of the translation significantly, however, the errors on the depth map got considerably worse. For the second approach we observed interchanged behavior. We therefore believe, that the estimation in Equation (6.3) is a good choice, as it balances the improvement for both errors.

To evaluate the result for the rotation, we compute the absolute error of the parametrization in the Lie algebra, i.e.

$$\delta_{\mathrm{w}} = \|\mathrm{w}_{true} - \mathrm{w}\|_2 \tag{6.4}$$

Recall that the length of w represents the rotation angle around the axis defined by the direction of w. Therefore the error measure δ_{w} can be seen as a combined evaluation of the rotation angle and direction.

6.1.3 Initial Values

Since the focus of this thesis is to analyze the performance of the prox-linear approach for the joint pose and depth optimization we neglect the estimation of initial values as discussed in Section 5.2.4. Instead, we are corrupting the ground truth and use the result to initialize the prox-linear method. We are combining three different models to do so.

Fringing of depth discontinuities To simulate miss-matching of pixels at object boundaries, we are fringing the discontinuities of the true depth map. To do so we select n_{fr} seed pixels at the discontinuities and set a random number of connected pixels within a distance of r_{fr} to the depth value of the respective seed pixel. The distance is measured in the maximum norm. This step is done before adding the noise described in the following.

Gaussian Noise We are using Gaussian noise on both the ground truth of pose and depth map. For the depth and the translation we use an individual standard deviation for each value, where we compute the standard deviation as a fraction of the value. That way we account for the possibly big range of those values and

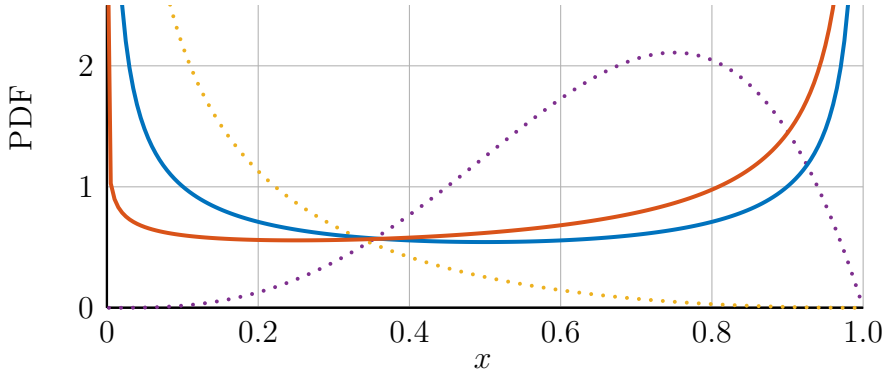


Figure 16: *Probability density functions of the beta distribution for different shape parameters. For the blue pdf* $\alpha = 0.4$ *and* $\beta = 0.4$ *have been chosen (note that it is symmetric), and for the red one* $\alpha = 0.8$ *and* $\beta = 0.4$*. For completeness two pdfs of Beta distributions with shape parameters bigger than 1 are shown as dotted lines. For yellow* $\alpha = 0.4$ *and* $\beta = 3$ *and for the purple one* $\alpha = 4$ *and* $\beta = 2$ *have been used.*

improve comparability between experiments on different scenes. For the exponential coordinates of the rotation we choose a fixed value for the standard deviation, since no big range is to be expected.

Hence, the initial values are obtained by the following equations

$$\begin{aligned} (\mathrm{w}_0)_i &= (\mathrm{w}_{true})_i + \varepsilon_{\mathrm{w}} \quad \text{with} \quad \varepsilon_{\mathrm{w}_i} \sim \mathcal{N}\left(\mu_{\mathrm{w}}, \sigma_{\mathrm{w}}^2\right) \\ (T_0)_i &= (T_{true})_i + \varepsilon_{T_i}, \quad \text{with} \quad \varepsilon_{T_i} \sim \mathcal{N}\left(\mu_T, (\xi_T \cdot (T_{true})_i)^2\right) \\ (\mathbf{h}_0^{vec})_i &= (\mathbf{h}_{true}^{vec})_i + \varepsilon_{h_i}, \quad \text{with} \quad \varepsilon_{h_i} \sim \mathcal{N}\left(\mu_h, (\xi_h \cdot (\mathbf{h}_{true}^{vec})_i)^2\right) \end{aligned} \tag{6.5}$$

where the ξ_* are the fractions for the computation of standard deviations for the respective quantities and the mean values μ_* represent possible biases.

Beta Distribution to Model Outliers The purely Gaussian noise might not be enough to model the behavior of the initial values that would be estimated in complete SLAM system accurately. Therefore we will add outliers to the initial depth map, which we model using a beta distribution with shape parameters $\alpha, \beta \in (0, 1)$. For $x \in [0, 1]$ the pdf of the beta distribution reads

$$f_{beta}(x) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha, \beta)}, \quad \text{with } B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} \tag{6.6}$$

where B is a normalization factor for which the Gamma function is used. The pdf for different pairs of shape parameters is shown in Figure 16.

The outlier model affects a certain percentage $p_{outliers}$ of the values in the depth map where the indices are chosen randomly. For those indices we add scaled beta random variables to the Gaussian noise.

$$(\mathbf{h}_0^{vec})_i = (\mathbf{h}_{true}^{vec})_i + \varepsilon_{h_i} + \xi_{outliers} \cdot (\mathbf{h}_{true}^{vec})_i (2\nu_i - 1) \tag{6.7}$$

with $\nu_i \sim Beta(\alpha, \beta)$ for $\alpha, \beta \in (0, 1)$ and ε_{h_i} as before. The factor $\xi_{outliers}$ is the percentage of the true value of the depth map that the outlier can deviate at most from this true value.

Fringing	
n_{fr}	3000
r_{fr}	5

Gaussian Noise	
μ_{w}	0
μ_T	0
μ_h	0
σ_{w}	2°
ξ_T	0.15
ξ_h	0.1

Outliers	
α	0.4
β	0.4
$\xi_{outliers}$	0.5
$\rho_{outliers}$	0.3

Table 2: *Standard parameters used for the generation of the initial values in the experiments.*

Values Used for the Experiments Unless stated otherwise, we use the parameter values in Table 2 for the corruption of the ground truth to obtain the initial values. Note that for repeatability we do not use biases on the pose. However, we did tests with random nonzero values to test the algorithm for sensitivity towards pose biases, where no such sensitivity could be observed.

As discussed in Section 3.5, we cannot recover the correct scale if the initialization is on the wrong scale. Therefore, to simplify the evaluation of the results, we do not impose a bias on the depth map and use a symmetric Beta distribution.

6.2 Performance of the Joint Optimization

In contrast to many other methods, we are estimating the pose and the depth map jointly instead of sequentially. In this section we evaluate the performance of this approach by comparing the results of the joint optimization to the estimation of only the pose and only the depth map, given the ground truth of the respective other quantity. For the estimation of only one part of the optimization variable, we modify our approach such that we use the ground truth whenever we would use the respective part of the optimization variable. Moreover, we need to reduce the Jacobian matrix accordingly. We use the standard parameters given in Section 6.1.1 for both the joint and the pure estimation of only one part of the optimization variable.

We evaluate the performance on both data sets. For the New Tsukuba data, it proved unnecessary to use any occlusion detection, which also makes the interpretation of the warped images in the results easier. For the Carla sequence we use a basic occlusion detection as described in Section 3.1.2. If two warped points are closer than 0.4 (pixels) in the maximum norm, we consider both of them invalid.

6.2.1 Performance Experiment New Tsukuba

The image pair used for this experiment is shown in Figure 17. Figure 18 shows the ground truth of the depth map and the initial value for the depth map used to initialize the algorithm. As described in Section 6.1.3, we obtain the initial value by corrupting the ground truth. The warped intensity resulting from the initial values can be seen on the left in Figure 19. The image shows for each pixel in the reference image the interpolated brightness value at the location of the pixel center warped to the second image. To obtain a sharp result, we removed the blurring of the image, which is used in the algorithm. That way, the warped image is easier to interpret.

Figure 17: *Image Pair used for the Performance Experiment New Tsukuba in Section 6.2.1. The reference image is on the right, the second image is on the left.*

The resulting depth map of the joint estimation is shown in Figure 20, which also includes a plot of the signed relative errors for the pixels, as described in Section 6.1.2. The depth map is corrected by the estimated scale factor $\hat{\rho}$ for the computation of the relative errors, as described in Section 6.1.2. The estimated scale factor is plotted in Figure 22 over the iterations. The warped image of the result of the joint estimation is shown on the right of Figure 19.

The result of the pure depth map estimation is shown in Figure 21. Since the scale drift does not occur in the estimation if the translation vector is known, the depth map is not corrected for the calculation of the relative errors.

To compare the resulting depth map of the joint approach to the result of the depth only approach, Figure 23 shows the percentage of pixels for which it holds for the relative error that $\delta_{\mathbf{h}_1} > 15\%$ for both the joint and the pure depth estimation.

The comparison of the result for the pose of the joint approach to the pure pose estimation is done in Figure 24. The figure shows the absolute errors δ_{w} on the left and the relative error δ_T of the translation on the right. Note that the translation for the joint optimization is corrected by the estimated scale factor, while the translation for the pure pose optimization is not. To get a feeling for the impact of the scaling factor on δ_T, the figure also shows the translation error without the scale factor correction.

6.2.2 Discussion of Performance on the New Tsukuba Dataset

The algorithm performs well on the New Tsukuba dataset, and the results of the joint optimization are comparable to the results of the separate optimizations. In the depth maps barely any difference is visible between the result of the joint approach and of the pure depth estimation. In both cases, discontinuities remain sharp, and even small details like the rods of the camera stand are preserved. The fringing of the edges at the object boundaries, on the other hand, could not be recovered in both cases. Since the prox-linear term limits the progress on the optimization variable in each step, and recovering the fringed edges requires correction by a big margin, this problem is likely due to the prox-linear approach. By considering the image pair in Figure 19, we see that, except for the fringed edges, the reconstruction from the photometric data term is excellent.

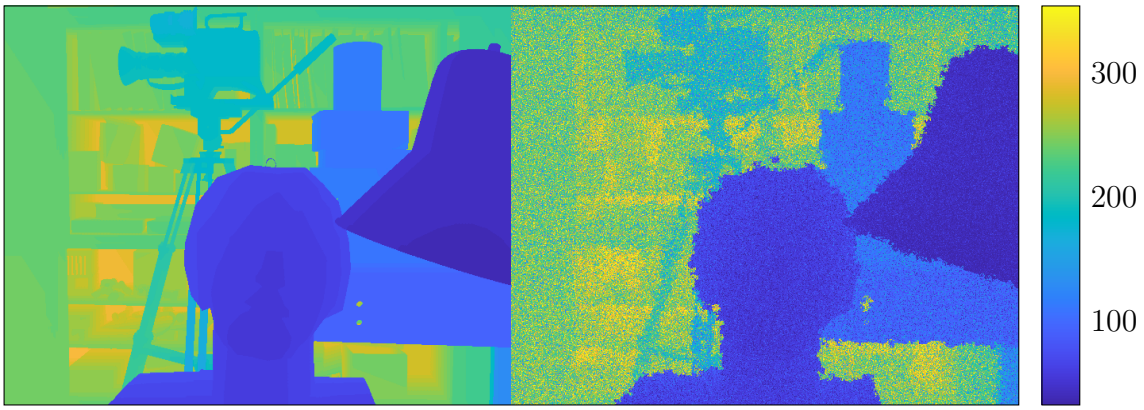


Figure 18: *On the left: The true depth map for the Performance Experiment New Tsukuba in Section 6.2.1. On the right: The initial value obtained by using the noise model described in Section 6.1.3.*

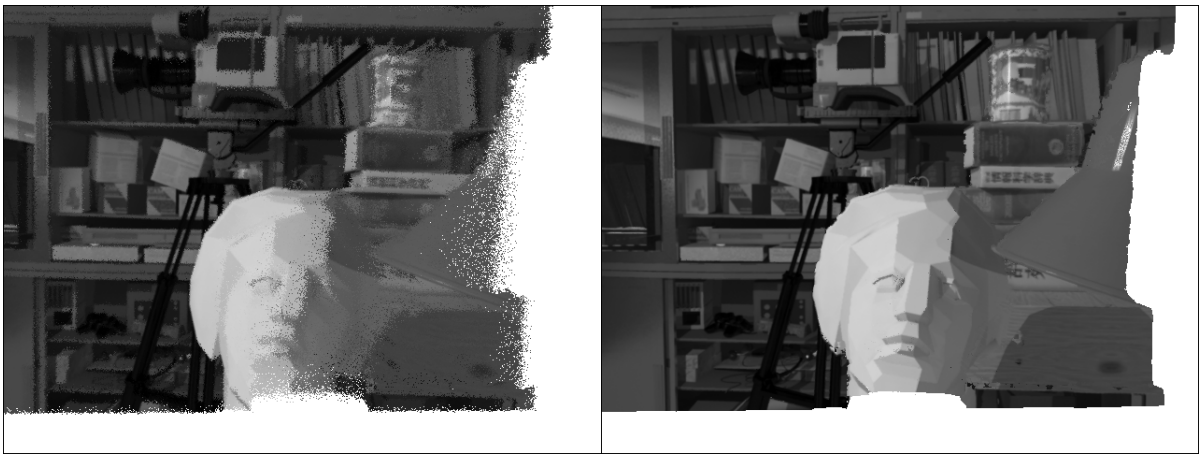


Figure 19: *Warped image for initial values on the left, and for the the result of the joint estimation on the right. The images show for each pixel in the reference frame the interpolated brightness value at the warped location in the second image. For the warping using the inital values the blurring was removed to see the correspondences more clearly. White pixels indicate invalidity, the black frame was added to indicate the image boundaries. Image pair for the Performance Experiment New Tsukuba in Section 6.2.1 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization.*

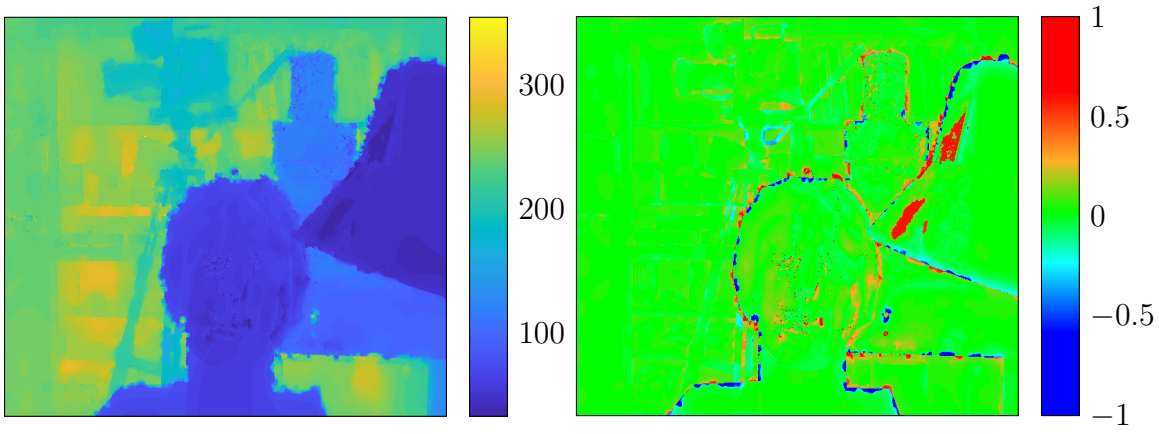


Figure 20: *Resulting depth map of the joint approach in the Performance Experiment New Tsukuba in Section 6.2.1 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth. Note that for the error the resulting depth map was corrected by the estimated scale factor.*

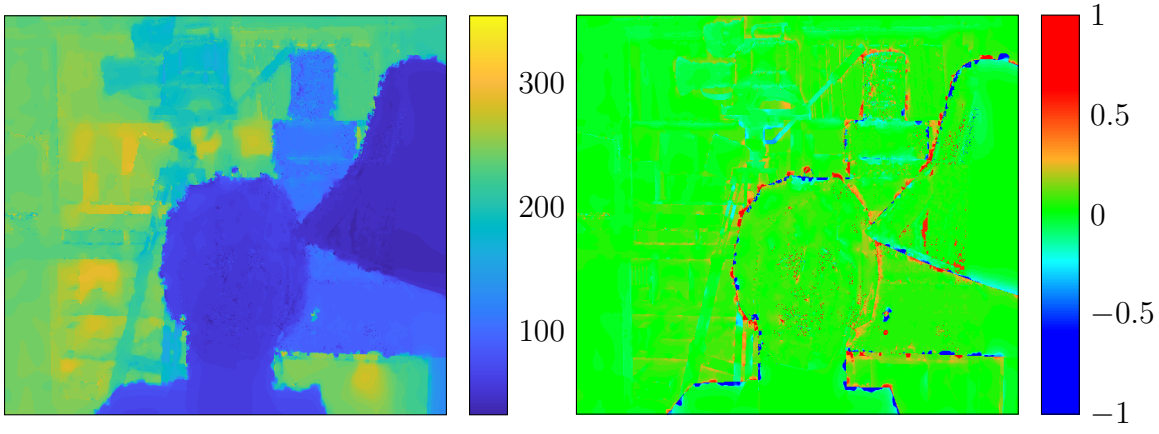


Figure 21: *Result of pure depth optimization in the Performance Experiment New Tsukuba in Section 6.2.1 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth. Note that for the relative error the depth map is unscaled.*

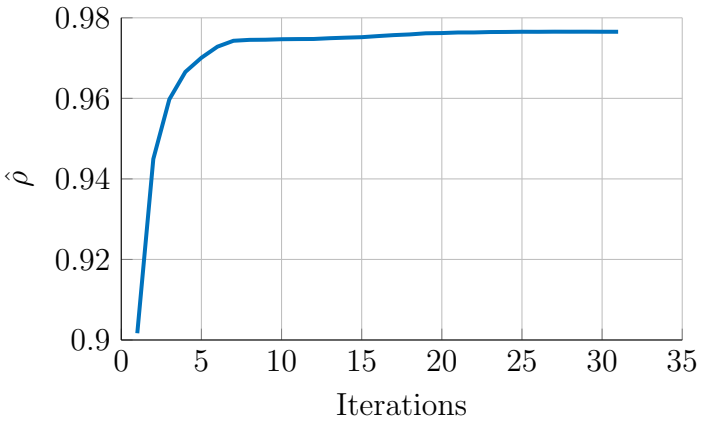


Figure 22: *Estimation of the scale factor for the joint approach in the Performance Experiment New Tsukuba in Section 6.2.1 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization.*

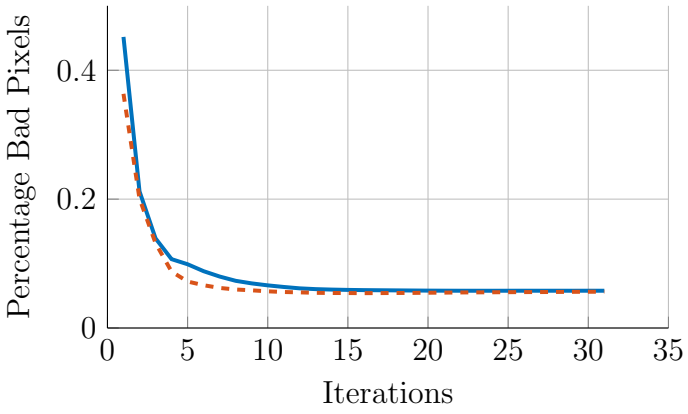


Figure 23: *Percentage of pixels with a relative error $\delta_{\mathbf{h}_i}$ of the estimated depth bigger than 15% plotted over the iterations for the Performance Experiment New Tsukuba in Section 6.2.1 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. The result for the joint estimation is shown in blue, the result of the pure depth estimation in orange. The depth map of the joint optimization is rescaled by $\hat{\rho}$ for the computation of the relative errors.*

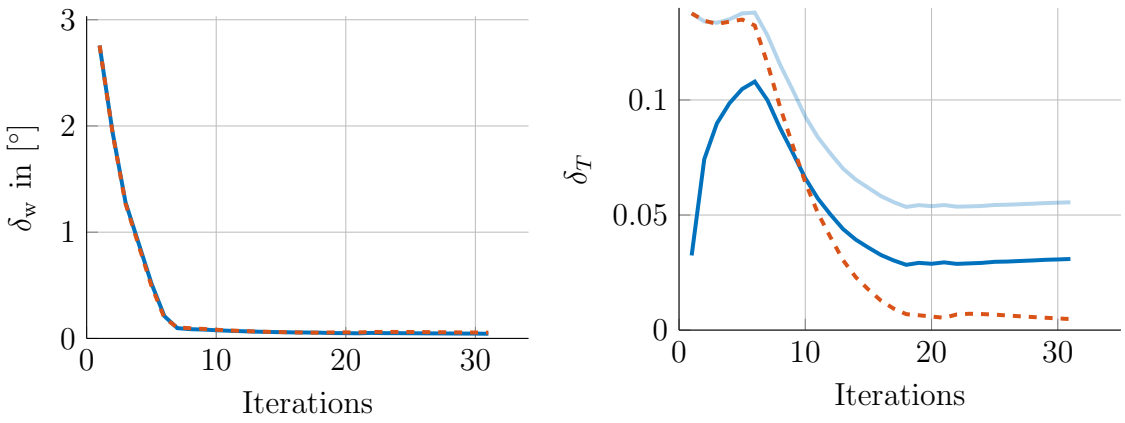


Figure 24: *Comparison of the pose estimation for the Performance Experiment New Tsukuba in Section 6.2.1 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. On the left the absolute errors between the parametrizations in the Lie algebra are shown, and on the right the relative errors of the translation vectors. The result for the joint estimation is shown in dark blue, the result of the pure pose estimation in orange. The translation vector of the joint optimization is rescaled by $\hat{\rho}$ for the computation of δ_T. For comparison, the relative error of the unscaled translation is shown in light blue.*

For the joint approach, two areas of significant false depth estimations are present on the lampshade. One possible explanation of why we encounter this problem in the joint approach but not in the pure approach is that those pixels started drifting during the initial phase before the pose was recovered. In that phase, we have an abundance of miss-matches due to the false pose in the joint approach. Consequently, the energy of the data term is much higher than the energy of the regularization term and therefore drifting of individual pixels is not stopped by regularization. This is in contrast to the pure approach, in which the amount of miss-matching is much lower, due to the correct pose. The drifting in the region of the lampshade is easier than elsewhere as it is a big uniform colored area in the image that is prone to miss-matches. In combination with the forward movement of the camera, this means that the pixels still get mapped to the lampshade in the second image, even if the depth values drift towards the camera.

The recovering of the rotation is very good and matches the estimation of the pure pose estimation. The estimation of the translation in the joint is less good than the result of the pure pose approach. Still, the error reduces significantly, which we see in particular by considering δ_T for the uncorrected translation. (Recall that the scale factor estimation in the first steps is not very reliable, and therefore the error for the scaled translation is not either). In Figure 22, we see that no scale drift seems to be present in this experiment, which confirms the assumption in Section 3.5, at least for this experiment.

We did perform experiments on a lot more image pairs of the data set, mostly with results comparable to the one presented here. For the quality of the depth map, barely any difference is observable between the joint approach and the pure depth estimation. Merely, occasional drifts of small regions of pixels, as described in this experiment have been encountered. For the recovery of the pose, the joint approach consistently performed a little worse than the pore pose estimation, in particular

on the translation. However, as discussed before, the relative error for the pose is highly sensitive towards the scale estimation, and therefore we need to treat these results with caution. Still, it can be said that the relative error in the joint approach was reduced significantly in most of the cases, and therefore the pose was recovered, at least to a certain extent.

We did encounter occasional false pose estimations, which affected both the joint as well as the pure pose approach. The main reason for these problems seems to be the following. The data contains image sequences, where the camera performs a forward movement only, with barely any rotation or lateral movement. Since, basically, the method is based on triangulation, the lack of lateral movement results in "narrow triangles" that make the pose estimation very hard.

Also, we did observe an occasional over-smoothing of the depth map in the background, while the foreground was still noisy. A possible explanation might be the direct parametrization of the depth. As mentioned in the remark in Section 3.4, Civera et al. reported superiority of the inverse depth parametrization for scenes with a large range of depth values.

For the comparison with the pure approaches, we need to keep in mind that these approaches are optimal, in that the ground truth is available. A realistic system based on a two-step approach would first estimate the pose, and then use this pose estimate for the depth estimation. Therefore we expect the results of real systems to be slightly worse than the results of the pure pose estimation used here.

6.2.3 Performance Experiment Carla

The image pair from the Carla sequence used in this experiment is shown in Figure 25. Now, Figure 26 shows the true depth map as well as the initial value. Note that the depth values for the sky are set to a maximum value of 10000 in the original data. However, for clarity, we clip the colormap for this section, as shown. Figure 27 shows the warped image of the first step and of the result of the joint approach where now the influence of the occlusion detection is visible by the many invalid pixels.

The rest of the results is presented in figures analogously to the experiment on the New Tsukuba data set. The estimated depth map of the joint approach is shown in Figure 28, while the result of the pure depth map estimation is shown in Figure 29. Figure 30 show the estimated scale factor over the iterations. To ensure a stable estimation, we needed to restrict the depth values used for the computation of $\hat{\rho}$ to values corresponding to the foreground, which was done using a threshold value. Figure 31 shows the comparison of the percentage of the bad pixels, and finally Figure 32 analyzes the results of the pose estimation.

6.2.4 Discussion of Performance on the Carla Dataset

We see that the algorithm performs a little less good on the Carla sequence. However, the results of the joint approach are still comparable to the separate optimizations. For the depth map, we encounter the same problems for the fringed edges, while still a good preservation of discontinuities is achieved.

At the board fence in front of the house, we can see problems with the basic occlusion detection we are using. The fence is prone to occlusion in particular in combination with the lateral movement which the camera performs. Therefore, in this area of the image, many pixels are set to invalid, as is visible in the warping of

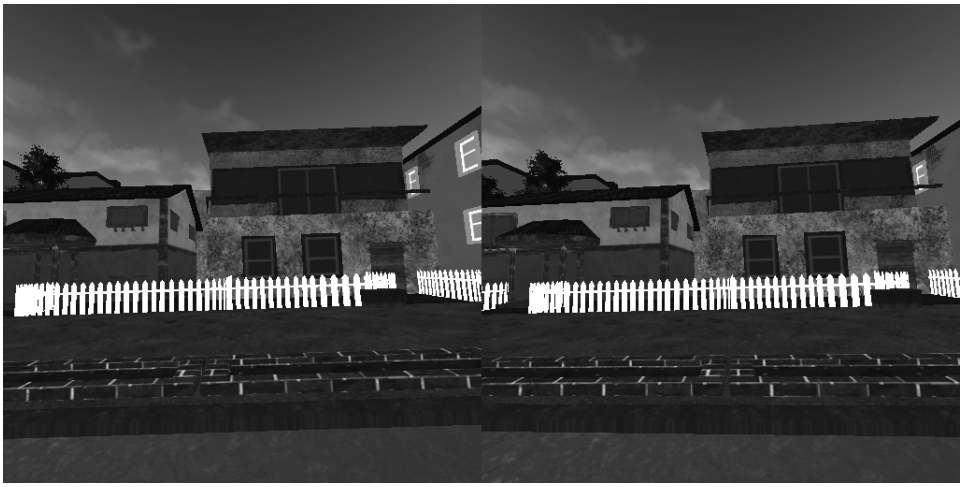


Figure 25: *Image Pair used for the Performance Experiment Carla in Section 6.2.3. The reference image is on the right, the second image is on the left.*

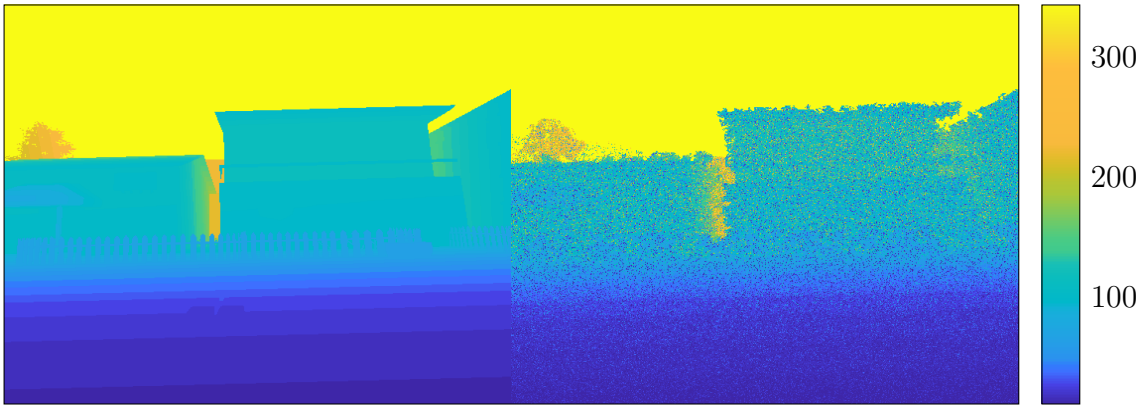


Figure 26: *On the left: The true depth map for the Performance Experiment Carla in Section 6.2.3. On the right: The initial value obtained by using the noise model described in Section 6.1.3.*

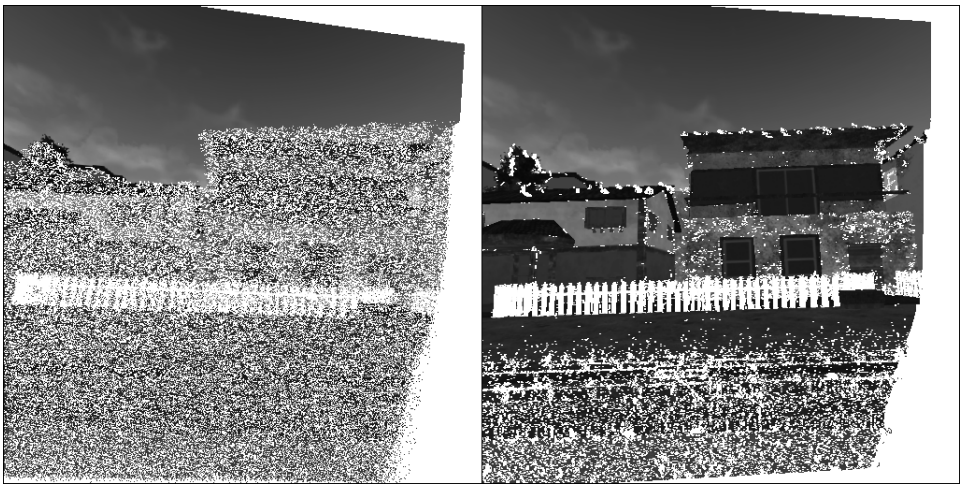


Figure 27: *Warped image for initial values on the left, and for the the result of the joint estimation on the right. The images show for each pixel in the reference frame the interpolated brightness value at the warped location in the second image. For the warping using the inital values the blurring was removed to see the correspondences more clearly. White pixels indicate invalidity, the black frame was added to indicate the image boundaries. Image pair for the Performance Experiment Carla in Section 6.2.3 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization.*

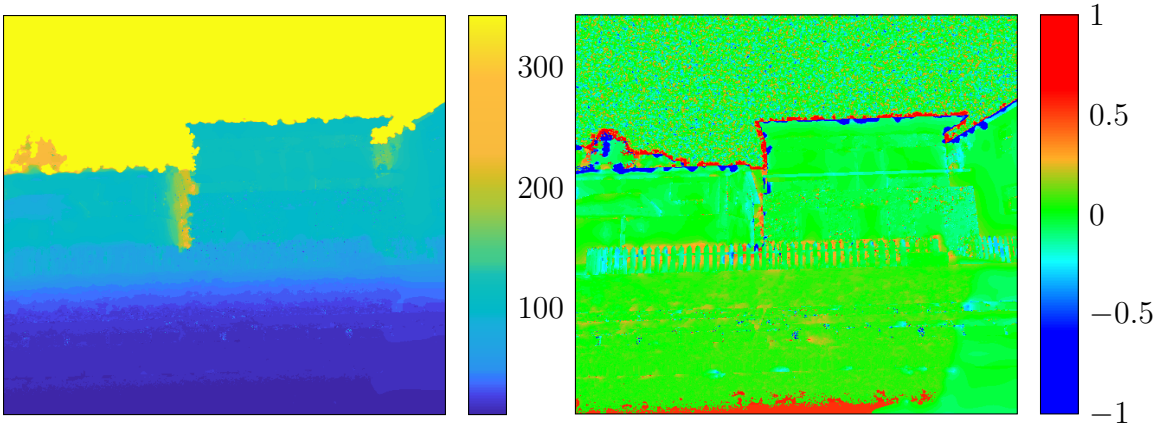


Figure 28: *Resulting depth map of the joint approach in the Performance Experiment Carla in Section 6.2.3 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth. Note that for the error the resulting depth map was corrected by the estimated scale factor.*

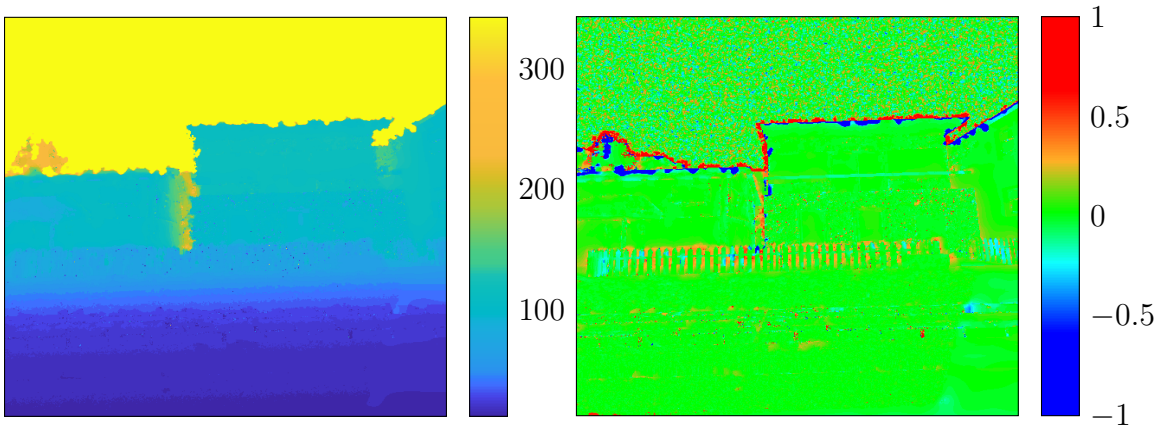


Figure 29: *Result of pure depth optimization in the Performance Experiment Carla in Section 6.2.3 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth. Note that for the relative error the depth map is unscaled.*

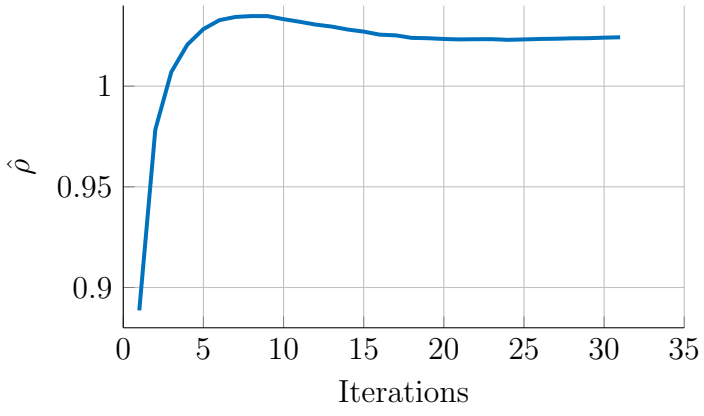


Figure 30: *Estimation of the scale factor for the joint approach in the Performance Experiment Carla in Section 6.2.3 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization.*

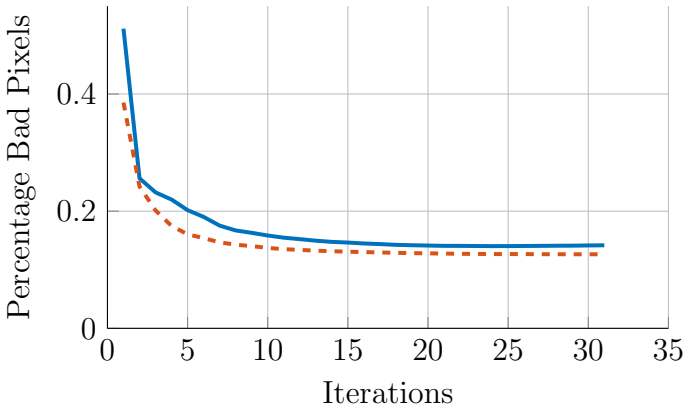


Figure 31: *Percentage of pixels with a relative error $\delta_{\mathbf{h}_i}$ of the estimated depth bigger than 15% plotted over the iterations for the Performance Experiment Carla in Section 6.2.3 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. The result for the joint estimation is shown in blue, the result of the pure depth estimation in orange. The depth map of the joint optimization is rescaled by $\hat{\rho}$ for the computation of the relative errors.*

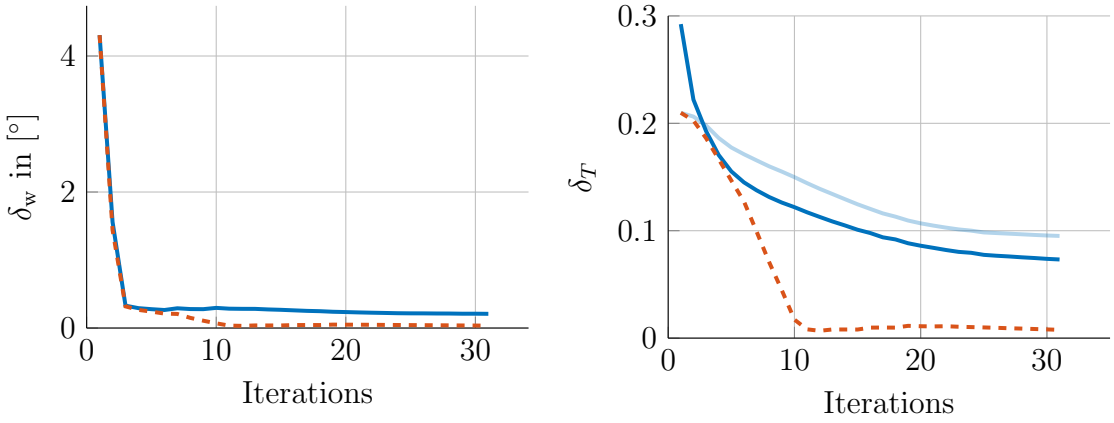


Figure 32: *Comparison of the pose estimation for the Performance Experiment Carla in Section 6.2.3 using the standard parameters in combination with the absolute data loss and isotropic Huber regularization. On the left the absolute errors between the parametrizations in the Lie algebra are shown, and on the right the relative errors of the translation vectors. The result for the joint estimation is shown in dark blue, the result of the pure pose estimation in orange. The translation vector of the joint optimization is rescaled by $\hat{\rho}$ for the computation of δ_T. For comparison, the relative error of the unscaled translation is shown in light blue.*

Regularization Term	
λ_{reg}	1.5
h_{reg}	1

Prox-Linear Algorithm	
ζ_{step}	0.95
$M_{0,\mathrm{w}}$	$3 \cdot 10^{-6}$
$M_{0,T}$	$1 \cdot 10^{-7}$
$M_{0,\mathbf{h}}$	1
$M_{min,\mathrm{w}}$	$2 \cdot 10^{-8}$
$M_{min,T}$	$5 \cdot 10^{-7}$
$M_{min,\mathbf{h}}$	$4 \cdot 10^{-1}$

Table 3: *Adjusted parameters for the special case discussed in Section 5.4 using a quadratic data loss*

the image in the last step in Figure 27 as well. Since invalid pixels disable the data term, this area is dominated by the regularization, which therefore over-smoothes the depth map. This can be seen in Figure 28, where the boards of the fence are too far behind, and the background is too close. This effect may be prevented by using a more sophisticated occlusion detection like the one described in Section 3.1.2.

In contrast to the New Tsukuba example, we find that the rotation is recovered a little less good than for the pure estimation. However, the result is still satisfactory. For the translation, we see that the initial value is improved significantly. However, the difference to the pure pose estimation is bigger than for the New Tsukuba example.

In general, the approach did not perform completely satisfactory on the Carla data set. While we were still able to recover a satisfactory pose and depth estimates in many cases, we observed more false pose estimates, and the overall quality of the estimation seemed to be less good than for the New Tsukuba data set. One possible reason for the reduced performance could be the lack of fine detail and the abundance of flat surfaces in the Carla scenery, which makes it harder for the data term to match pixels. Due to the flat surfaces, it might help to employ the generalized total variation regularization as described in Section 3.3.1. Another problem for the algorithm might be the vast uniform colored surfaces or the surfaces like the wall of the house, which are prone to miss-matches of the data term.

Also, we observed more problems with the basic occlusion detection approach. Especially for thin separated structures like traffic light posts, the well-known phenomenon of the structures appearing doubled in the warped picture occurred. Hence, for scenes containing such structures, a more sophisticated occlusion detection method is advisable.

6.3 Comparison of Data Loss Functions

In this section, we compare the results obtained by using different data loss functions. Moreover, we inspect the convergence properties of the approaches. The reference approach is the absolute loss function with the standard parameters, that already has been examined in Section 6.2. For the second choice, we replace the absolute loss by the Huber loss with Huber parameter $h_{loss} = 15$. The rest of the standard parameters remain unchanged. The third approach is the special case discussed in Section 5.4 in which we use the quadratic loss. For simplicity, we often refer to this approach simply by *quadratic loss* in this section.

Figure 33: *Image Pair used for the Comparison of Data Loss Functions in Section 6.3. The reference image is on the right, the second image is on the left.*

Since we are linearizing the regularization term in the third approach, we need to adjust the parameters. The changes to the standard parameters (only for the third approach) are shown in Table 3. Note that we had to reduce the initial step widths considerably, in particular for the depth map, to stabilize the optimization. The parameters for the diagonal damping term remained unchanged and are only given for completeness.

Remark. We did try to use the quadratic data loss in the general prox-linear setting that was used for the absolute and the Huber loss. However, the primal-dual splitting, as described in Section 5.3, seems to be a poor choice in this case, and we have experienced major convergence issues. With the approach described inSection 5.4, we encountered much less problems. Moreover, this approach is significantly faster than the other two, which allows the comparison based on another criterion.

Due to the reduced step widths for the depth in the third approach, the smoothing of the depth map is significantly slower. For that reason, and to be able to assess the long-time behavior of the approaches, we increase the number of linearizations for all three cases to 250.

Since we are evaluating the convergence of the algorithm via the energy, we need to disable the iterative decrease of the blurring and use a fixed blurring of $\sigma_{Gauss} = 3$ instead. Otherwise, the energy of the data term would jump every time the blurring is reduced because the sharpened images require a more precise matching. This behavior is preventing a clear assessment of the energies and does not occur for a fixed blurring. Note that this affects the sharpness of details in the resulting depth map.

For comparison, we did measure the time required to solve the sub-problems for the different approaches. While the focus of the implementation is not on performance, the timing still gives a rough idea of what to expect in terms of run time.

The image pair used for this experiment is shown in Figure 33, the true depth map and the initial value in Figure 34. The resulting depth maps of the three approaches are shown in Figures 35 and 36 and Figure 37, respectively. Figure 38 shows the estimated scale factors, Figure 39 the percentages of bad pixels and in Figure 40 the evaluation of the pose is plotted. Figure 41 shows the energies, where besides the total energy also the data and the regularization proportion is plotted. Finally, Figure 42 shows the computation times for the different approaches.

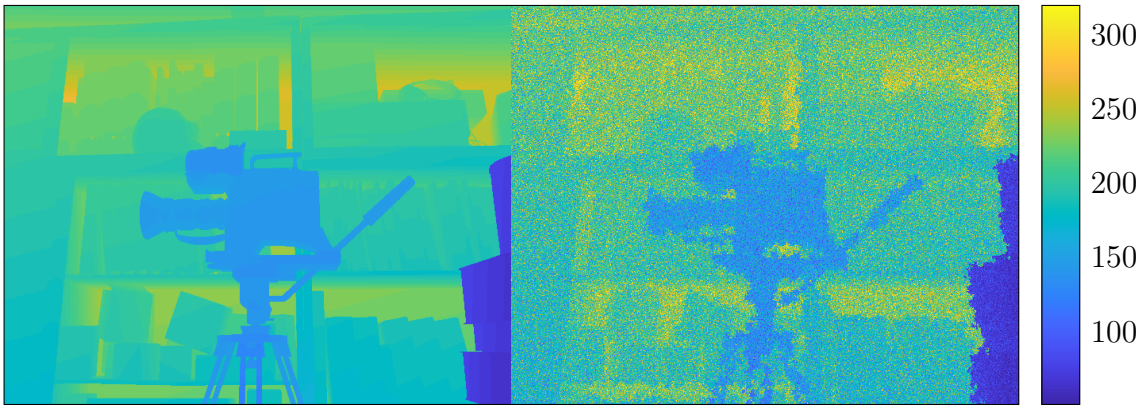


Figure 34: *On the left the ground truth of the depth map for the Comparison of Data Loss Functions in Section 6.3. On the right the initial value obtained by using the noise model described in Section 6.1.3.*

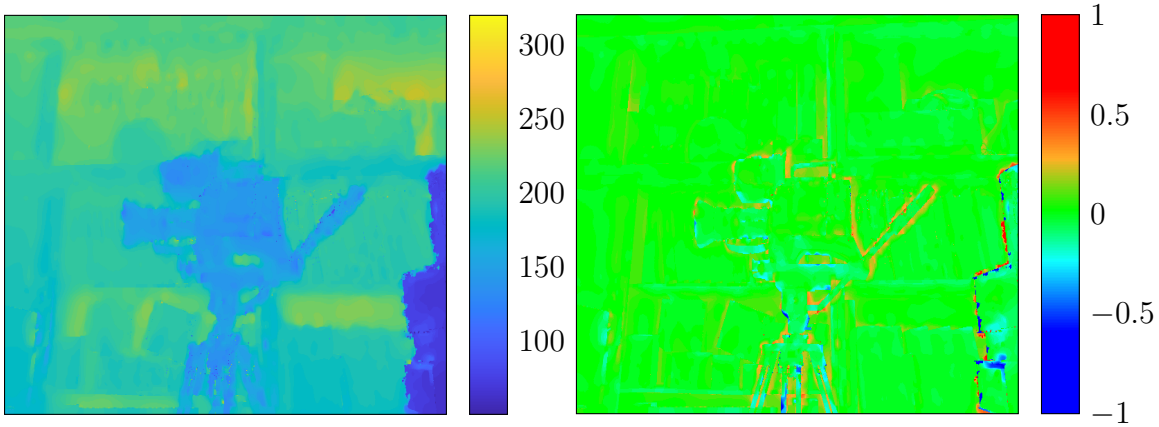


Figure 35: *Result depth map for the absolute loss data term in the Comparison of Data Loss Functions in Section 6.3 using the standard parameters in combination with the isotropic Huber regularization. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth.*

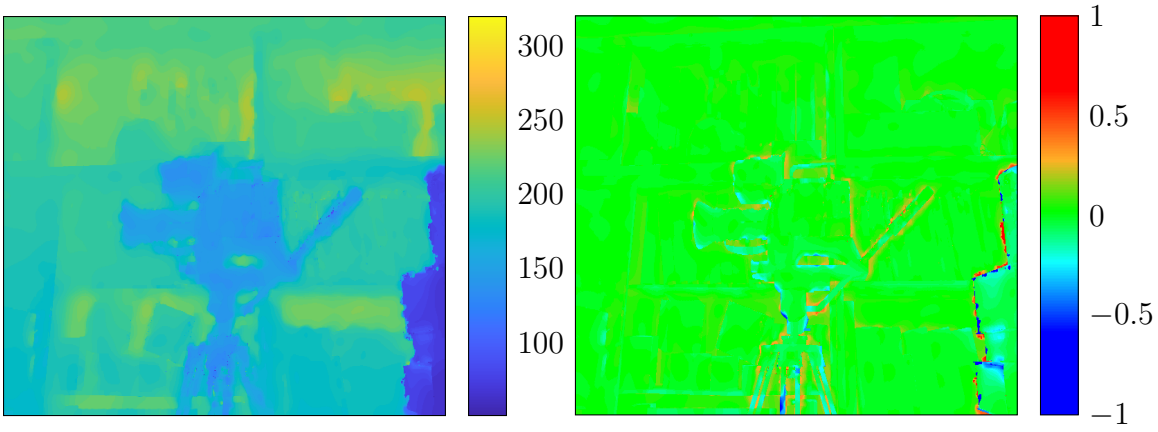


Figure 36: *Result depth map for the Huber loss data term in the Comparison of Data Loss Functions in Section 6.3 using the standard parameters in combination with the isotropic Huber regularization. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth.*

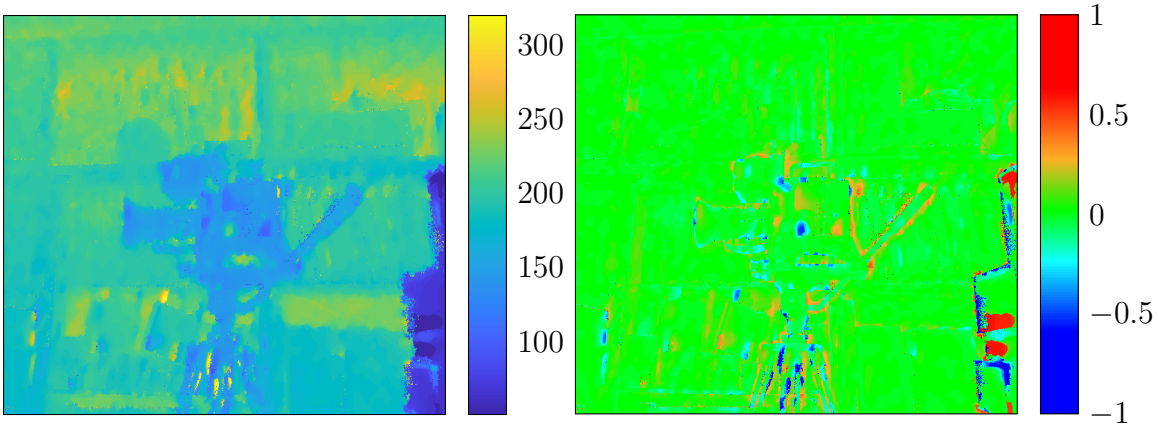


Figure 37: *Result depth for the special case discussed in Section 5.4 which uses a quadratic data loss and a repeatedly linearized isotropic Huber regularization. Figure for the Comparison of Data Loss Functions in Section 6.3. On the left the depth map is shown, on the right the signed relative error of this depth map with respect to the ground truth.*

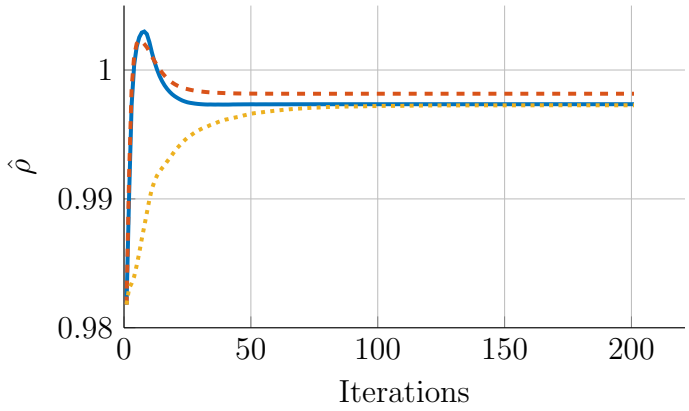


Figure 38: *Estimation of the scale factors in the Comparison of Data Loss Functions in Section 6.3. The result for the absolute data loss with standard parameters and isotropic Huber regularization is shown in blue, the result for the Huber loss with standard parameters and isotropic Huber regularization in orange and the result for the the special case from Section 5.4 with the quadratic loss and the repeatedly linearized isotropic Huber regularization is shown in yellow.*

Discussion

First, we consider the quality of the resulting depth maps. Between the absolute and the Huber loss, barely any difference is noticeable. The result obtained using the Huber loss seems to be a little smoother, and the relative errors seem to be slightly smaller, but the difference is minor. This impression is confirmed when we consider Figure 39, where the percentage of the bad pixels is a little lower.

The special case using the quadratic loss, on the other hand, is a little less smooth. While we preserve discontinuities in the depth map as well, the noisy input is still showing slightly, in particular, in the background. The possible reason for this is twofold. First, the regularizing abilities of the data term might be reduced considerably due to the linearization. Second, due to the small initial step width for the depth ($M_{0,\mathbf{h}} = 1$) and the constant step width reduction, the step width is too small to allow for any further progress after a certain amount of iterations. This problem affects the background in particular since the noise level is higher in this region (recall that the noise level is coupled to the true depth value). We did try to increase the initial step width on the depth. However, this destabilized the algorithm considerably. Possibly a different approach for the step width reduction might help to increase the performance for this approach. Alternatively, the inverse depth parametrization might again bring an improvement, since it significantly reduces the range of the depth values.

All approaches perform equally well on the recovery of the rotation. On the pose, a difference is noticeable, and the absolute loss clearly performs best. Since the estimated scale factor for the special case is very close to the factor estimated for the absolute loss, we can assume that the special case indeed performed a little bit worse than the absolute loss. For the Huber loss, on the other hand, the difference in the estimated scale factor makes a comparison a bit tricky. Possibly the same translation result would be better with another estimation. Still, we see that all approaches improve the initial translation value.

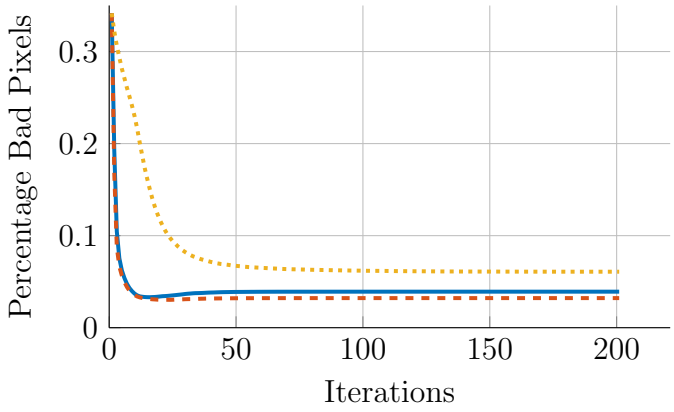


Figure 39: *Percentage of pixels with a relative error of the estimated depth bigger than 15% plotted over the iterations for the Comparison of Data Loss Functions in Section 6.3. The result for the absolute data loss with standard parameters and isotropic Huber regularization is shown in blue, the result for the Huber loss with standard parameters and isotropic Huber regularization in orange and the result for the the special case from Section 5.4 with the quadratic loss and the repeatedly linearized isotropic Huber regularization is shown in yellow. For all relative errors the estimated scale $\hat{\rho}$ has been used.*

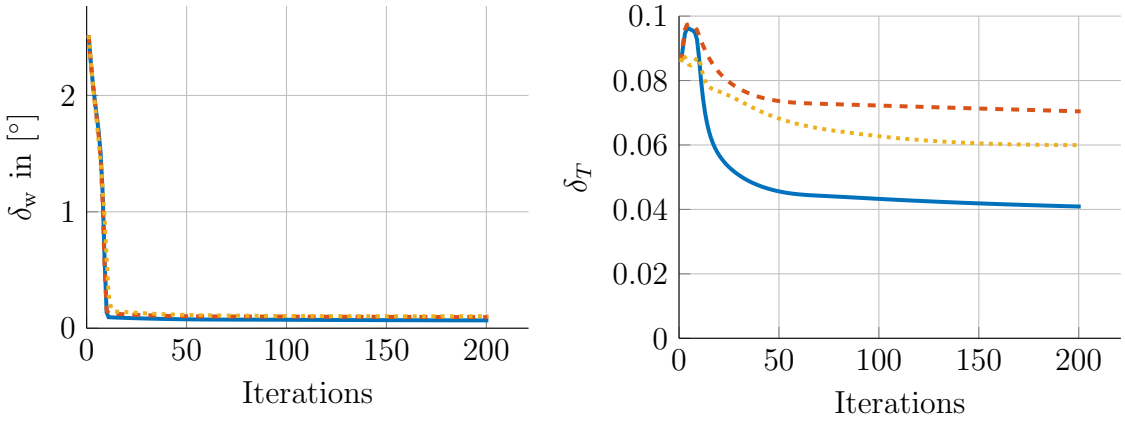
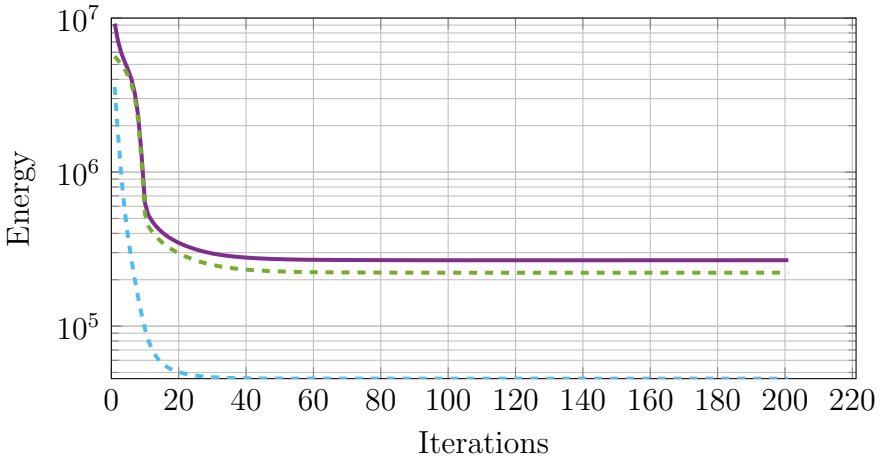
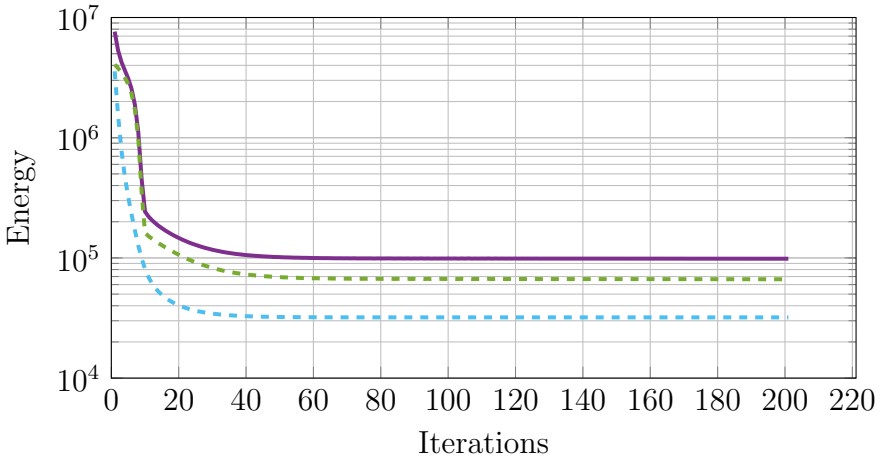


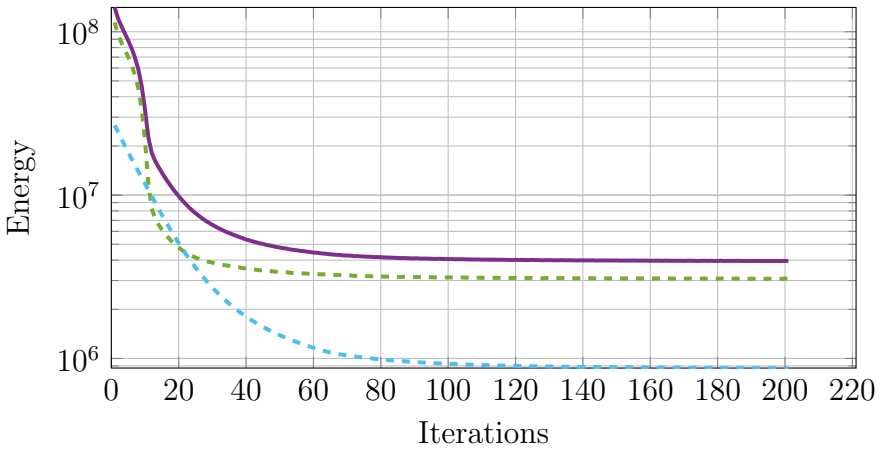
Figure 40: *Comparison of the pose estimation for the Comparison of Data Loss Functions in Section 6.3 over the iterations. The result for the absolute data loss with standard parameters and isotropic Huber regularization is shown in blue, the result for the Huber loss with standard parameters and isotropic Huber regularization in orange and the result for the the special case from Section 5.4 with the quadratic loss and the repeatedly linearized isotropic Huber regularization is shown in yellow. The translation vector is rescaled by $\hat{\rho}$ for the computation of the error in all cases.*



(a)



(b)



(c)

Figure 41: *Energies for the different data loss functions in in the Comparison of Data Loss Functions in Section 6.3. For all plots the total energy is shown in purple, the data part in cyan and the regularization part is shown in green. (a) Absulute data loss with standard parameters and isotropic Huber regularization (b) Huber data loss with standard parameters and isotropic Huber regularization(c) Special case from Section 5.4 with the quadratic loss and the repeatedly linearized isotropic Huber regularization*

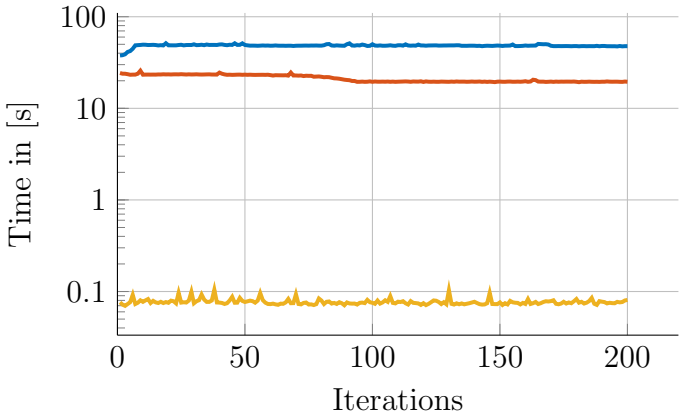


Figure 42: *Computation times for the different data loss functions in in the Comparison of Data Loss Functions in Section 6.3. The result for the absolute data loss with standard parameters and isotropic Huber regularization is shown in blue, the result for the Huber loss with standard parameters and isotropic Huber regularization in orange and the result for the the special case from Section 5.4 with the quadratic loss and the repeatedly linearized isotropic Huber regularization is shown in yellow.*

From Figure 41, we see that we have a monotonic decrease of the energy for all three cases. This shows that we have chosen step widths that are small enough. Recall that the approach to the step widths is purely heuristic, and no rigorous proofs for the prox-linear approach with individual step widths have been done. Therefore, this result is a necessary justification for the approach as a whole.

Again, we did additional experiments with multiple image pairs. The results are similar to the result of the experiment presented here. The quality of the depth maps recovered by the absolute and the Huber loss is usually quite similar and in general a bit better than the quality of the depth map obtained by the special case based on the quadratic loss. For the quality of the pose, no significant differences between the approaches have been found. Occasional false pose estimates affected all the approaches, where again the lack of lateral movement in the images seemed to be a major reason. The quadratic loss function in the third approach proved to be surprisingly robust, considering that it is prone to be affected by outliers. However, the good performance might also be due to the model chosen for the creation of the initial values and the relatively low percentage of outliers (0.3).

Finally, we consider the computation times in Figure 42. As we would expect, solving the subproblem via the Schur complement for the third approach is significantly faster than the other approaches. Especially with real-time applications in mind, this is a huge advantage, if the slightly reduced quality of the depth map is acceptable. Note that the parameters for all approaches are not chosen to maximize performance. The number of iterations, in particular, is set rather big to ensure to capture most of the relevant behavior. In a real system, it is likely possible to reduce those number to improve the computation times. Also, the use of GPUs for the highly parallelizable PDHG approach is possible to improve the efficiency further.

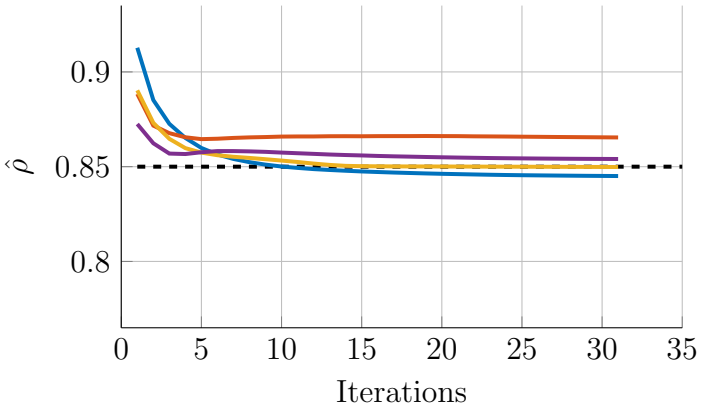


Figure 43: *Estimated scale factors for the first set of experiments in the Analysis of the Scale Factor Estimation in Section 6.4. The ground truth was scaled by* $\frac{1}{\rho}$ *where* $\rho = 0.85$ *was chosen. For the estimation the standard parameters with absolute loss and isotropic Huber regularization have been used.*

6.4 Analysis of the Scale Factor Estimation

In this section, we show that the estimator of the scale factor yields a good estimation $\hat{\rho}$. While this is not directly related to the analysis of the SLAM approach, it still helps to build confidence in the error measures used to evaluate the results.

We use again the standard parameters defined in Section 6.1.1 with the absolute data loss and the isotropic Huber Regularization. Also, we use the standard noise parameters given in Section 6.1.3 to create the initial values. However, before corrupting the ground truth, we scale both the depth map and the translation by $\frac{1}{\rho}$, which creates a scaling error.

Consider the definition of the relative errors with the scale factor estimate. We see that if we do not drift away from the initial wrong scale, the optimal scale correction factor is $\hat{\rho} = \rho$ which reverts the creation of the scaling error.

In a first experiment we choose $\rho = 0.85$. We apply the algorithm to 4 different image pairs of the New Tsukuba data set and consider the estimated scale factors, which are shown in Figure 43. We see that the estimates for all experiments converge towards the optimal scale factor, which is shown as a dotted black line.

We repeat the experiment for the choice $\rho = 1.3$. The estimated scale factors are shown in Figure 44. Both experiments show that the scale factor estimator detects errors in the scale reliably.

6.5 Evaluation of Scale Drift

In Section 3.5, we claimed that the potential scale drift was no problem in practice, thanks to the damping of the vanishing step width. In this section, we evaluate this assumption. We use the standard parameters with the absolute data loss and the isotropic Huber regularization again, and perform several experiments on distinct image pairs of the New Tsukuba data set.

To ensure that we do not end the algorithm early, we increase the number of linearizations to 350. With the standard parameters, this means that the step width for the depth has decreased to $50 \cdot 0.9^{350} \approx 4.8 \cdot 10^{-15}$ at the end. Since, for such a

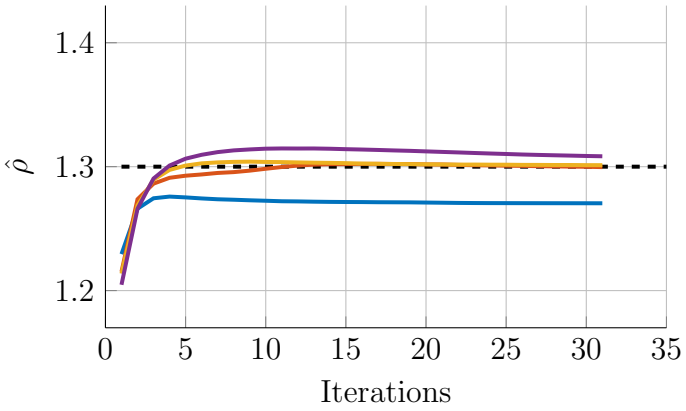


Figure 44: *Estimated scale factors for the second set of experiments in the Analysis of the Scale Factor Estimation in Section 6.4. The ground truth was scaled by* $\frac{1}{\rho}$ *where* $\rho = 1.30$ *was chosen. For the estimation the standard parameters with absolute loss and isotropic Huber regularization have been used.*

small step width, almost no further progress is possible, we are sure that we capture the complete behavior of the algorithm.

The obvious choice to evaluate if scale drift is present is to use the scale factor estimator $\hat{\rho}$, which we introduced in Section 6.1.2. Although we did see in Section 6.4 that $\hat{\rho}$ provides a reliable estimation of the scaling factor, we consider two more quantities to ensure, that we do not miss any drift by using an unsuitable measure. On the one hand, we use the mean depth; on the other hand, we use the norm of the translation vector. We expect both quantities to change with the (potential) scale drift.

We perform multiple experiments where we use a scaling error on the initial values for some of them, analogously to Section 6.4. This is done to investigate if the initialization on a wrong scale triggers a scale drift.

The results are shown in Figures 45 and 46 and Figure 47. For none of the quantities under consideration, we did find any indication of scale drift. This suggests that the chosen step width parameters provide a good trade-off that allows enough flexibility for the estimation in the beginning, and ensures fast enough reduction to prevent any drift. Note that this statement is coupled to the noise level. For higher noises, it is possible that the reduction is too fast and prevents the complete recovery, of the depth map in particular.

Remark. We did observe scale drift problems during the tuning of the parameters. This suggests that the phenomenon is indeed present and that a good choice for the step width parameters is necessary to prevent it. With the final set of parameters, we were unable to re-produce any scale drift any more, despite extensive testing.

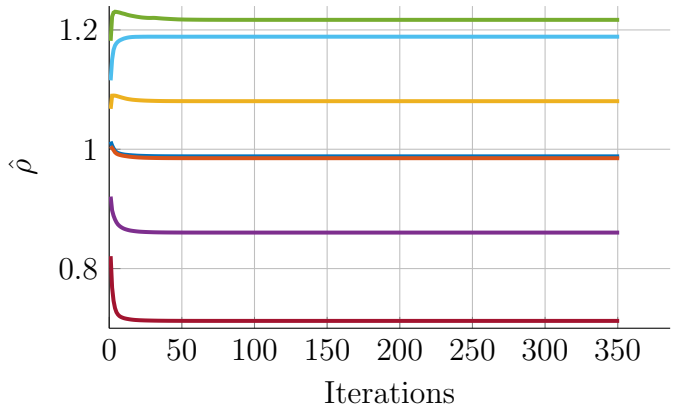


Figure 45: *Estimated scale factors for the experiments in the Evaluation of Scale Drift in Section 6.5. For the experiments we used the standard parameters in combination with the absolute data loss and the isotropic Huber regularization.*

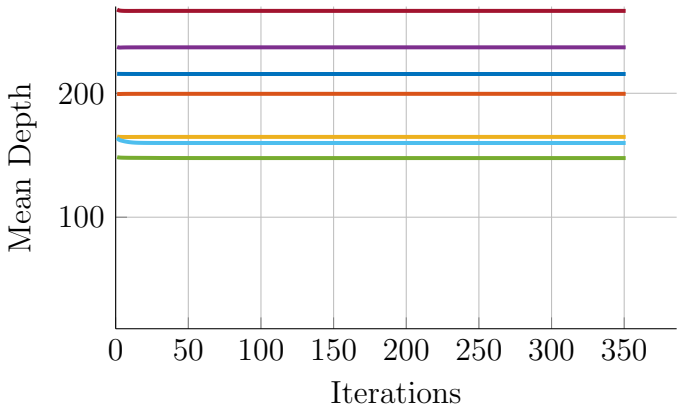


Figure 46: *Mean depths for the experiments in the Evaluation of Scale Drift in Section 6.5. For the experiments we used the standard parameters in combination with the absolute data loss and the isotropic Huber regularization.*

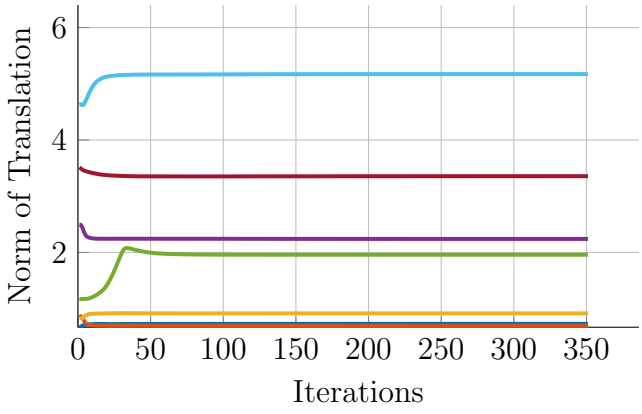


Figure 47: *Norm of the translation vectors for the experiments in the Evaluation of Scale Drift in Section 6.5. For the experiments we used the standard parameters in combination with the absolute data loss and the isotropic Huber regularization.*

7 Conclusion

In this thesis, we examined a dense photometric SLAM method for the joint optimization of the pose and the depth map. For the solution of the non-linear and non-convex optimization problem, we proposed a modification of the prox-linear algorithm by using individual step widths for the components of the optimization variable. The convergence of the algorithm was shown experimentally, and the algorithm proved well suited for the optimization.

The results of the experiments show that the joint approach is valid and that both quantities can be estimated simultaneously. While the results did not quite reach the quality of the pure pose or depth estimation, we have to keep in mind that those pure approaches can rely on the ground truth. In future work it would be interesting to compare the performance of the approach presented in this thesis to other state-of-the-art methods in order to have a more realistic comparison for our method.

Admittedly, the model for the generation of the initial values used in this thesis is ad-hoc and might not represent the input from a reals system well. Therefore, it is an exciting research perspective to integrate the approach into a full SLAM system, including the estimation of initial values. Besides the interaction of our approach with the initial values generated by such a system, there are several other challenges.

One of these challenges is real-time capability, which is a crucial requirement for many SLAM systems. While the timing results presented in this thesis do not seem very promising at first glance, we have not exploited the full potential yet. The main improvement is to be expected from implementing the algorithm on a GPU to exploit the possibility of parallelization. Additionally, it is likely, that the number of iterations can be reduced without compromising the quality of the results too much, which will improve efficiency further. Finally, the implementation in a more efficient programming language like C++ will also bring an improvement in computation time.

The results for the special case described in Section 5.4 are surprisingly promising. Although we use a non-robust data term and a repeatedly linearized regularization term rather than the original term, the results of this approach did not seem to be significantly worse than the results obtained using robust data loss and full linearization terms. When considering the computation times, the approach even has a considerable advantage. It would be interesting to investigate if the good performance is a result of the model for the input parameters, or if this approach performs equally well in a more realistic setting.

One major problem for the evaluation of the results is the estimation of the scale drift. While the results of the approach seemed to be good, errors in the scale estimation may have flawed that evaluation, in particular for the translation, which is highly dependent on the estimation. In this case, too, the integration into a real system is necessary for a better assessment. This also enables intriguing research on the integration of data from other sensors to contain the possibility of scale drift.

Another open question that is not necessarily related to the SLAM setting alone, is a rigorous analysis of the prox-linear approach with a weighted prox term. While the case of the standard prox term has been investigated intensively over the past years, we are not aware of any work on the case with individual step widths. Since

this approach proved quite useful for our setting, the derivation of optimal choices for the step widths, as well as a proof of convergence for this case, is an interesting question that requires future research.

A Basic Results Calculus

In the following, we will define the calculus notation layout that we are using in this thesis, and state two results.

Definition A.1 (Gradient)**.** Let $f : \mathbb{R}^n \to \mathbb{R}$ be a C^1 function. Then the gradient is defined by

$$\nabla f(y) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(y) \\ \vdots \\ \frac{\partial f}{\partial x_n}(y) \end{bmatrix}$$

If we want to emphasize the vector variable with respect to which the gradient is calculated we write the variable as a subscript, e.g. $\nabla_x f$.

Definition A.2 (Jacobian Matrix)**.** Let $f : \mathbb{R}^n \to \mathbb{R}^m$ be a C^1 function. Then the Jacobian matrix is defined by

$$Jf(y) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1}(y) & \dots & \frac{\partial f_1}{\partial x_n}(y) \\ \vdots & & \vdots \\ \frac{\partial f_m}{\partial x_1}(y) & \dots & \frac{\partial f_m}{\partial x_n}(y) \end{bmatrix} = \begin{bmatrix} \nabla f_1{}^T(y) \\ \vdots \\ \nabla f_m{}^T(y) \end{bmatrix}$$

If we want to emphasize the vector variable with respect to which the Jacobian matrix is calculated, we write the variable as a subscript, e.g. $J_x f$.

Lemma A.1 (Multivariate Chain Rule - Case 1)**.** Let $g : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}$ be C^1 functions. Then the gradient of the composition is given by

$$\nabla f(g(y)) = Jg(y)^T \nabla f(g(y))$$

Lemma A.2 (Multivariate Chain Rule - Case 2)**.** Let $g : \mathbb{R}^n \to \mathbb{R}^m$ and $f : \mathbb{R}^m \to \mathbb{R}^k$ be C^1 functions. Then the Jacobian matrix of the composition is given by

$$J(f \circ g)(y) = Jf(g(y))\, Jg(y)$$

Remark. Note that the layout convention defined here is inconsistent. While the gradient following Definition A.1 yields a column vector, the "Jacobian" matrix of a function $f : \mathbb{R}^n \to \mathbb{R}$ yields a row vector. This is the reason that we need the distinction between Lemma A.1 and Lemma A.2. We will, however, stick to the layout introduced here as it seems to be the most widespread and has advantages for the linearization.

B Supplementary Material for Implementation

B.1 Finite Difference Matrices

As discussed in Section 5.1 we can write the difference operators as matrices in combination with the vectorized depth maps. Recall that the vectorization of the

depth map was obtained by stacking the columns of $\mathbf{h}$. The matrix formulations of Definition 3.1 read

$$D_x = \begin{bmatrix} -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & -1 & 0 & & 0 & 1 & 0 & & \vdots & & \vdots \\ & \ddots & \ddots & \ddots & & \ddots & \ddots & \ddots & & & \\ & & 0 & -1 & 0 & & 0 & 1 & 0 & & \\ 0 & \cdots & 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \hline 0 & \cdots & 0 & & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ \vdots & & \vdots & & 0 & -1 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 \\ & & & & & \ddots & \ddots & & & \ddots & \ddots & & \\ & & & & & & & & & & & & \end{bmatrix} \begin{matrix} \\ \\ \\ \\ m \\ \\ m+1 \\ \\ \\ \end{matrix}$$

$$\begin{matrix} & & & & & & & m & m+1 & & \\ \end{matrix}$$

$$D_y = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ 0 & -1 & 1 & 0 & \cdots & 0 & & \vdots \\ \vdots & & \ddots & \ddots & & & & \\ 0 & \cdots & 0 & -1 & 1 & 0 & & \\ 0 & \cdots & & & 0 & 0 & \cdots & 0 \\ \hline 0 & \cdots & & & 0 & -1 & 1 & 0 \\ \vdots & & & & & & \ddots & \ddots \\ & & & & & 0 & & \end{bmatrix} \begin{matrix} \\ \\ \\ \\ m \\ \\ m+1 \\ \\ \end{matrix}$$

Marked in blue is the non-zero part of the matrix that is necessary to calculate the gradients for the first column of the image. The gradient operator is continued for the next column of the image by shifting this matrix by m to the right and attaching the shifted matrix at the bottom.

Note that D_x needs to be appended by m zero rows at the end, as we cannot compute finite x-differences on the right boundary of the image with the chosen boundary conditions. While this does not add any information, the rows are necessary to keep a consistent dimension of the operators.

B.2 Image Gradient at the Warped Location

For the linearization we need the image gradient $\nabla \mathcal{J} I_i \left({}_i\mathbf{x}_j\right)$ at the warped points ${}_i\mathbf{x}_j = \omega_{\mathbf{x}_j}\left(R\left(\mathrm{w_i}\right), T_i, \mathbf{h}\right)$. The respective formulas depend on the chosen interpolation method for the image.

Bilinear Interpolation In this case $\mathcal{J} I_i = \mathcal{J}_{lin} I_i$ is not differentiable. As stated in Section 5.2 we will use the bilinear interpolation of the image gradient at the pixel centers to approximate the image gradient at the warped locations. The image gradient at the pixel centers is obtained by central differences to avoid offsets. For $I_i \in \mathbb{R}^{m \times n}$ and a pixel center $\mathbf{x} = (x, y) \in \mathcal{P}$ the central differences are defined as

$$D_x^c I_i\left(\mathbf{x}\right) = \begin{cases} \frac{1}{2}\left(I_i\left(x+1, y\right) - I_i\left(x-1, y\right)\right) & \text{if } 1 < x < n \\ 0 & \text{else} \end{cases}$$

$$D_y^c I_i\left(\mathbf{x}\right) = \begin{cases} \frac{1}{2}\left(I_i\left(x, y+1\right) - I_1\left(x, y-1\right)\right) & \text{if } 1 < y < m \\ 0 & \text{else} \end{cases}$$

By bilinear interpolation of the discrete gradients at the pixel centers we obtain the approximation of the image gradient at the warped location

$$\nabla_{\mathbf{x}} \left(\mathcal{J}_{lin} I_i \right) (\mathbf{x}) = \begin{bmatrix} \mathcal{J}_{lin} \left(D_x^c I_i \right) (\mathbf{x}) \\ \mathcal{J}_{lin} \left(D_y^c I_i \right) (\mathbf{x}) \end{bmatrix} \tag{B.1}$$

Bicubic Interpolation In this case $\mathcal{J} I_i = \mathcal{J}_{cub} I_i$ is differentiable. We can calculate

$$\frac{\partial \mathcal{J}_{cub} I_i}{\partial x} (\mathbf{x}) = \sum_l \sum_k I_{kl} W' (x - x_k) W (y - y_k)$$
$$\frac{\partial \mathcal{J}_{cub} I_i}{\partial y} (\mathbf{x}) = \sum_l \sum_k I_{kl} W (x - x_k) W' (y - y_k)$$

where

$$W'(x) = \frac{\partial W}{\partial x}(x) = \begin{cases} 3 \operatorname{sgn}(x) (a+2) x^2 - 2 (a+3) x & \text{for } |x| \leq 1 \\ \operatorname{sgn}(x) (3ax^2 + 8a) - 10ax & \text{for } 1 < |x| < 2 \\ 0 & \text{else} \end{cases} \tag{B.2}$$

B.3 Derivation of the Prox Operators

To be able to implement the PDHG algorithm efficiently we need closed form solutions for the prox operators of G and F^* which we will discuss in this chapter. Recall that $F^* = F_1^* + F_2^*$, so we first need expressions for the convex conjugate of $F_1(x) = h(x - b)$ for the different loss functions h as well as for $F_2(x) = \lambda_{reg} \mathcal{R}(x)$. Having derived those convex conjugates we can compute the prox operators.

B.3.1 Closed-Form Solutions for Convex Conjugates

We first state several lemmas that we will need for the derivations

Lemma B.1. Let $E, \tilde{E} : \mathbb{R}^n \to \overline{\mathbb{R}}$, $b \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ such that

$$E(u) = \alpha \tilde{E}(u - b)$$

Then

$$E^*(p) = \langle p, b \rangle + \alpha \tilde{E}^* \left(\frac{p}{\alpha} \right)$$

Proof. We directly calculate

$$\begin{aligned} E^*(p) &= \sup_{u \in \mathbb{R}^n} \left(\langle p, u \rangle - \alpha \tilde{E}(u - b) \right) \\ &= \alpha \sup_{u \in \mathbb{R}^n} \left(\langle \frac{p}{\alpha}, b \rangle + \langle \frac{p}{\alpha}, u - b \rangle - \tilde{E}(u - b) \right) \\ &= \langle p, b \rangle + \alpha \sup_{u \in \mathbb{R}^n} \left(\langle \frac{p}{\alpha}, u - b \rangle - \tilde{E}(u - b) \right) \end{aligned}$$

By doing a change of variables and defining $v := u - b$ we can write

$$\begin{aligned} E^*(p) &= \langle p, b \rangle + \alpha \sup_{v \in \mathbb{R}^n} \left(\langle \frac{p}{\alpha}, v \rangle - \tilde{E}(v) \right) \\ &= \langle p, b \rangle + \alpha \tilde{E}^* \left(\frac{p}{\alpha} \right) \end{aligned}$$

which finishes the proof. □

Lemma B.2. Let $F_1 : \mathbb{R}^{n_1} \to \overline{\mathbb{R}}$ and $F_2 : \mathbb{R}^{n_2} \to \overline{\mathbb{R}}$ and define $F : \mathbb{R}^{n_1+n_2} \to \overline{\mathbb{R}}$ as $F(u_1, u_2) = F_1(u_1) + F_2(u_2)$. Then

$$F^*(p) = F_1^*(p_1) + F_2^*(p_2), \quad \text{with } p = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix}$$

Proof. We calculate

$$\begin{aligned} F^*(p) &= \sup_q \langle p, q \rangle - F(q) \\ &= \sup_{q_1, q_2} \langle p_1, q_1 \rangle + \langle p_2, q_2 \rangle - F_1(q_1) - F_2(q_2) \\ &= \sup_{q_1} \langle p_1, q_1 \rangle - F_1(q_1) + \sup_{q_2} \langle p_2, q_2 \rangle - F_2(q_2) \\ &= F^*(p_1) + F_2^*(p_2) \end{aligned}$$

□

Lemma B.3. Let $\|\cdot\|$ be any norm on $\mathbb{R}^n$. Then for $f(x) = \|x\|$ the convex conjugate reads

$$f^*(p) = \delta_{\|\cdot\|_* \leq 1}(p) = \begin{cases} 0 & \text{if } \|p\|_* \leq 1 \\ \infty & \text{else} \end{cases}$$

where $\|\cdot\|_*$ is the dual norm of $\|\cdot\|$ defined by

$$\|p\|_* = \sup \{\langle u, p \rangle \mid \|u\| \leq 1\} = \sup \left\{ \frac{\langle u, p \rangle}{\|u\|} \mid \|u\| \neq 0 \right\}$$

Proof. See e.g. [4, Example 3.26] □

Convex Conjugates of Basic Norms We give the convex conjugates of some basic norms which we will use in the light of Lemma B.1 for the later proofs.

Proposition B.1. Let $E(u) = \frac{1}{2}\|u\|_2^2$. Then the convex conjugate reads

$$E^*(p) = \frac{1}{2}\|p\|_2^2$$

Proof. We calculate

$$E^*(p) = \sup_q \langle p, q \rangle - \frac{1}{2}\|q\|_2^2$$

From the optimality conditions it is straightforward that the supremum is obtained at $\hat{q} = p$. Hence the convex conjugate is $E^*(p) = \langle p, p \rangle - \frac{1}{2}\|p\|_2^2 = \frac{1}{2}\|p\|_2^2$. □

Proposition B.2. Let $E(u) = \|u\|_1$. Then the convex conjugate reads

$$E^*(p) = \delta_{\|\cdot\|_\infty \leq 1}(p) = \begin{cases} 0 & \text{if } \|p\|_\infty \leq 1 \\ \infty & \text{else} \end{cases}$$

Proof. Using Lemma B.3 we only need to show that the infinity norm is indeed the dual norm of the 1-norm, i.e. we need to show that $\sup\{\langle u,p\rangle \mid \|u\|_1 \leq 1\} = \|p\|_\infty$. We estimate the scalar product

$$\langle u,p\rangle = \sum_{i=1}^{n} u_i p_i \leq \left|\sum_{i=1}^{n} u_i p_i\right| \leq \sum_{i=1}^{n} |u_i p_i| \leq \|p\|_\infty \sum_{i=1}^{n} |u_i| = \|p\|_\infty \|u\|_1 \leq \|p\|_\infty$$

where we used the condition $\|u\|_1 \leq 1$ for the last estimate. The upper bound can be obtained by choosing u as a vector of zeros with a ± 1 at the index of the maximum element in p where we choose the sign such that we get a positive value. This shows the that the dual norm of the 1-norm is indeed the infinity norm which finishes the proof. □

Proposition B.3. Let $E(u) = \|u\|_2$. Then the convex conjugate reads

$$E^*(p) = \delta_{\|\cdot\|_2 \leq 1}(p) = \begin{cases} 0 & \text{if } \|p\|_2 \leq 1 \\ \infty & \text{else} \end{cases}$$

Proof. Using Lemma B.3 we need to show that the 2-norm is its own dual norm, i.e. we need to show that $\sup\{\langle u,p\rangle \mid \|u\|_2 \leq 1\} = \|p\|_2$. Using the Cauchy-Schwarz inequality we can estimate the scalar product

$$\langle u,p\rangle \leq \|u\|_2 \|p\|_2 \leq \|p\|_2$$

where we used the condition $\|u\|_2 \leq 1$ for the last estimate. Since the 2-norm of p can be obtained we have showed the claim □

Proposition B.4. Let $E(u) = \|u\|_h$ defined as in Definition 3.2. Then the convex conjugate reads

$$E^*(p) = \frac{h}{2}\|p\|_2^2 + \delta_{\|\cdot\|_2 \leq 1}(p) = \frac{h}{2}\|p\|_2^2 + \begin{cases} 0 & \text{if } \|p\|_2 \leq 1 \\ \infty & \text{else} \end{cases}$$

Proof. As shown in [2, Chapter 6.7] we can write the Huber norm as the infimal convolution[7]of the Euclidean norm with the function $\omega_h(x) = \frac{1}{2h}\|x\|_2^2$. Hence, by defining $f(x) = \|x\|_2$ we can write

$$E(u) = (f \square \omega_h)(x)$$

As shown in [2, Chapter 4.5] the convex conjugate of a infimal convolution is the sum of the conjugates of the convoluted functions. Therefore we can write

$$E^*(p) = f^*(p) + \omega_h^*(p)$$

The conjugate of f reads

$$f^*(p) = \delta_{\|\cdot\|_2 \leq 1}(p) = \begin{cases} 0 & \text{if } \|p\|_2 \leq 1 \\ \infty & \text{else} \end{cases}$$

To show this we use Lemma B.3 and prove that the 2-norm is its own dual norm, i.e. that $\sup\{\langle u,p\rangle \mid \|u\|_2 \leq 1\} = \|p\|_2$. We bound the scalar product using the Cauchy-Schwarz-Inequality

$$\langle u,p\rangle \leq \|u\|_2 \|p\|_2 \leq \|p\|_2$$

where the condition of the dual norm for the last inequality. By choosing $u = \frac{p}{\|p\|_2}$ we obtain the upper bound which shows the claim.

The conjugate of ω_h can be obtained using Lemma B.1 and Proposition B.1. We write

$$\omega_h^*(p) = \frac{1}{h}\frac{1}{2}\|hp\|_2^2 = \frac{h}{2}\|p\|_2^2$$

□

Convex Conjugates for F_1 and F_2 We now can state the convex conjugates for the functions F_1 and F_2, including all the different possibilities.

Proposition B.5 (Convex Conjugate of Quadratic Loss Dataterm)**.** Let $b \in \mathbb{R}^n$, $\alpha > 0$ and consider $F : \mathbb{R}^n \to \mathbb{R}$ with $F(x) = \frac{\alpha}{2}\sum_i (x_i - b_i)^2$. Then the convex conjugate reads

$$F^*(p) = \langle p, b\rangle + \frac{1}{2\alpha}\|p\|_2^2$$

Proof. We write $F(x) = \alpha\tilde{F}(x - b)$ with $\tilde{F}(x) = \frac{1}{2}\|x\|_2^2$. Using Lemma B.1 and Proposition B.2 we can write

$$F^*(p) = \langle p, b\rangle + \alpha\tilde{F}^*\left(\frac{p}{\alpha}\right) = \langle p, b\rangle + \frac{1}{2\alpha}\|p\|_2^2$$

□

Proposition B.6 (Convex Conjugate of Absolute Loss Dataterm)**.** Let $b \in \mathbb{R}^n$, $\alpha > 0$ and consider $F : \mathbb{R}^n \to \mathbb{R}$ with $F(x) = \alpha\sum_i |x_i - b_i|$. Then the convex conjugate reads

$$F^*(p) = \langle p, b\rangle + \delta_{\|\cdot\|_\infty \le \alpha}(p)$$

Proof. We write $F(x) = \alpha\tilde{F}(x - b)$ with $\tilde{F}(x) = \|x\|_1$. Using Lemma B.1 and Proposition B.2 we can write

$$F^*(p) = \langle p, b\rangle + \alpha\tilde{F}^*\left(\frac{p}{\alpha}\right) = \langle p, b\rangle + \alpha\begin{cases}0 & \text{if } \left\|\frac{p}{\alpha}\right\|_\infty \le 1\\ \infty & \text{else}\end{cases}$$

As the condition $\|p/\alpha\|_\infty \le 1$ is equivalent to $\|p\|_\infty \le \alpha$ and we can omit the scaling factor for the indicator function, we can write

$$\begin{aligned}F^*(p) &= \langle p, b\rangle + \begin{cases}0 & \text{if } \|p\|_\infty \le \alpha\\ \infty & \text{else}\end{cases}\\ &= \langle p, b\rangle + \delta_{\|\cdot\|_\infty \le \alpha}(p)\end{aligned}$$

□

[7] The infimal convolution of two proper functions $h_1, h_2 : \mathbb{R}^n \to \overline{\mathbb{R}}$ is defined as

$$(h_1 \square h_2)(x) = \min_{u\in\mathbb{R}^n}\{h_1(u) + h_2(x - u)\}$$

Proposition B.7 (Convex Conjugate of Huber Loss Dataterm)**.** Let $b \in \mathbb{R}^n$ and $\alpha, h > 0$ and consider $F : \mathbb{R}^n \to \mathbb{R}$ with $F(x) = \alpha \sum_i |x_i - b_i|_h$. Then the convex conjugate reads

$$F^*(p) = \langle p, b \rangle + \frac{h}{2\alpha} \|p\|_2^2 + \delta_{\|\cdot\|_\infty \leq \alpha}(p)$$

Proof. With Lemma B.2 in mind we consider the function $f(x_i) = \alpha |x_i - b_i|_h$. Using Lemma B.1 and Proposition B.4 we obtain the convex conjugate as

$$f^*(p_i) = p_i b_i + \alpha \left(\frac{h}{2} \left(\frac{p_i}{\alpha} \right)^2 + \delta_{|\cdot| \leq 1} \left(\frac{p_i}{\alpha} \right) \right) = p_i b_i + \frac{h}{2\alpha} p_i^2 + \delta_{|\cdot| \leq \alpha}(p_i)$$

Note that the Euclidean norm im 1 dimension is reduced to the absolute function. Using Lemma B.2 we obtain the complete convex conjugate as

$$\begin{aligned} F^*(p) &= \sum_{i=1}^n p_i b_i + \frac{h}{2\alpha} p_i^2 + \delta_{|\cdot| \leq \alpha}(p_i) \\ &= \langle p, b \rangle + \frac{h}{2\alpha} \|p\|_2^2 + \delta_{\|\cdot\|_\infty \leq \alpha}(p) \end{aligned}$$

□

Proposition B.8 (Convex Conjugate of isotropic TV Regularization)**.** Let $\mathbf{\Gamma} \in \mathbb{R}_+^n$ and $\lambda_{reg} > 0$ and consider $F : \mathbb{R}^{2n} \to \mathbb{R}$ with

$$F(\mathbf{x}) = \lambda_{reg} \sum_{i=1}^n \Gamma_i \left\| x_{[i,i+n]} \right\|_2$$

Then the convex conjugate reads

$$F^*(p) = \delta_{\|\cdot\|_2 \leq \lambda_{reg} \Gamma_i} \left(p_{[i,i+n]} \right)$$

Proof. With Lemma B.2 in mind we consider the function $f(y) = \gamma_i \|y\|_2$ with the short form $\gamma_i = \lambda_{reg} \Gamma_i$. Using Lemma B.1 and Proposition B.3 we obtain the convex conjugate as

$$f^*(p) = \gamma_i \delta_{\|\cdot\|_2 \leq 1} \left(\frac{p}{\gamma_i} \right) = \gamma_i \delta_{\|\cdot\|_2 \leq \gamma_i}(p)$$

By using Lemma B.2 we obtain the complete convex conjugate as

$$F^*(p) = \delta_{\|\cdot\|_2 \leq \lambda_{reg} \Gamma_i} \left(\begin{bmatrix} p_i \\ p_{i+n} \end{bmatrix} \right)$$

□

Proposition B.9 (Convex Conjugate of isotropic Huber Regularization)**.** Let $\mathbf{\Gamma} \in \mathbb{R}_+^n$ and $\lambda_{reg}, h > 0$ and consider $F : \mathbb{R}^{2n} \to \mathbb{R}$ with

$$F(\mathbf{x}) = \lambda_{reg} \sum_{i=1}^n \Gamma_i \left\| x_{[i,i+n]} \right\|_h$$

Then the convex conjugate reads

$$F^*(p) = \sum_{i=1}^n \frac{h}{2\lambda_{reg} \Gamma_i} \left\| p_{[i,i+n]} \right\|_2^2 + \delta_{\|\cdot\|_2 \leq \lambda_{reg} \Gamma_i} \left(p_{[i,i+n]} \right)$$

Proof. With Lemma B.2 in mind we consider the function $f(y) = \gamma_i \|y\|_h$ with the short form $\gamma_i = \lambda_{reg}\Gamma_i$. Using Lemma B.1 and Proposition B.4 we obtain the convex conjugate as

$$f^*(p) = \gamma_i \left(\frac{h}{2} \left\| \frac{p}{\gamma_i} \right\|_2^2 + \delta_{\|\cdot\|_2 \leq 1} \left(\frac{p}{\gamma_i} \right) \right) = \frac{h}{2\gamma_i} \|p\|_2^2 + \delta_{\|\cdot\|_2 \leq \gamma_i}(p)$$

By using Lemma B.2 we obtain the complete convex conjugate as

$$F^*(p) = \sum_{i=1}^{n} \frac{h}{2\lambda_{reg}\Gamma_i} \left\| \begin{bmatrix} p_i \\ p_{i+n} \end{bmatrix} \right\|_2^2 + \delta_{\|\cdot\|_2 \leq \lambda_{reg}\Gamma_i} \left(\begin{bmatrix} p_i \\ p_{i+n} \end{bmatrix} \right)$$

□

B.3.2 Closed-Form Solutions for Prox Operators

Having derived the convex conjugates for the relevant functions, we can now compute the prox operators. While we are considering the weighted prox operators from Definition 4.2, we restrict the weighting matrices to diagonal matrices. If we have a diagonal matrix $M \in \mathbb{R}^{n \times n}$ we denote the i-th diagonal element with a slight abuse of notation by M_i.

We start with the prox operator of the function G.

Proposition B.10. Let $b \in \mathbb{R}^n$ and let $\Lambda, M \in \mathbb{R}^{n \times n}$ be positive definite diagonal matrices. Consider $G : \mathbb{R}^n \to \mathbb{R}$ with $G(u) = \frac{1}{2}\|u - b\|_M^2$. Then the weighted prox operator reads

$$\text{prox}_{\Lambda G}(x) = \left(\Lambda^{-1} + M^{-1}\right)^{-1} \left(\Lambda^{-1}x + M^{-1}b\right)$$

Proof. The preconditioned prox operator is defined as

$$\text{prox}_{\Lambda G}(x) = \hat{y} = \underset{y \in \mathbb{R}^n}{\arg\min} \left\{ \frac{1}{2}\|y - x\|_\Lambda^2 + \frac{1}{2}\|u - b\|_M^2 \right\}$$

Since the objective function is differentiable we can directly check the optimality conditions

$$\begin{aligned} & 0 = \Lambda^{-1}(\hat{y} - x) + M^{-1}(\hat{y} - b) \\ \Leftrightarrow \quad & \hat{y} = \left(\Lambda^{-1} + M^{-1}\right)^{-1} \left(\Lambda^{-1}x + M^{-1}b\right) \end{aligned}$$

□

Next we give the prox steps for the different loss functions.

Proposition B.11 (Prox Operator for Quadratic Loss Dataterm)**.** Let $b \in \mathbb{R}^n$, $\alpha > 0$ and let $\Sigma \in \mathbb{R}^{n \times n}$ be a positive definite diagonal matrix. Consider $F^* : \mathbb{R}^n \to \mathbb{R}$ with $F^*(p) = \langle p, b \rangle + \frac{1}{2\alpha}\|p\|_2^2$. Then the weighted prox operator reads

$$\text{prox}_{\Sigma F^*}(x) = \left(\Sigma^{-1} + \frac{1}{\alpha}I\right)^{-1} \left(\Sigma^{-1}x - b\right)$$

Proof. The preconditioned prox operator is defined as

$$\operatorname{prox}_{\Sigma F^*}(x) = \hat{y} = \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \left\{ \frac{1}{2} \|y - x\|_\Sigma^2 + \langle y, b \rangle + \frac{1}{2\alpha} \|y\|_2^2 \right\}$$

Since the objective function is differentiable we can directly check the optimality conditions

$$\begin{aligned} 0 &= \Sigma^{-1}(\hat{y} - x) + b + \frac{1}{\alpha}\hat{y} \\ \Leftrightarrow \quad \hat{y} &= \left(\Sigma^{-1} + \frac{1}{\alpha} I\right)^{-1} \left(\Sigma^{-1} x - b\right) \end{aligned}$$

□

Proposition B.12 (Prox Operator for Absolute Loss Dataterm)**.** Let $b \in \mathbb{R}^n$, $\alpha > 0$ and let $\Sigma \in \mathbb{R}^{n \times n}$ be a positive definite diagonal matrix. Consider $F^* : \mathbb{R}^n \to \mathbb{R}$ with $F^*(p) = \langle p, b \rangle + \delta_{\|\cdot\|_\infty \leq \alpha}(p)$. Then the componets of the weighted prox operator are given by

$$\left(\operatorname{prox}_{\Sigma F^*}(x)\right)_i = \begin{cases} (x - \Sigma b)_i & \text{if } |(x - \Sigma b)_i| \leq \alpha \\ \alpha \operatorname{sgn}(x - \Sigma b)_i & \text{else} \end{cases}$$

Proof. We re-write the optimization problem for the preconditioned prox operator

$$\begin{aligned} \operatorname{prox}_{\Sigma F^*}(x) &= \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \left\{ \frac{1}{2} \|y - x\|_\Sigma^2 + \langle y, b \rangle + \delta_{\|\cdot\|_\infty \leq \alpha}(y) \right\} \\ &= \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \left\{ \frac{1}{2} \|y - (x - \Sigma b)\|_\Sigma^2 + \delta_{\|\cdot\|_\infty \leq \alpha}(y) \right\} \end{aligned}$$

The last step is justified by the following considerations

$$\begin{aligned} \frac{1}{2} \|y - (x - \Sigma b)\|_\Sigma^2 &= \frac{1}{2} \langle \Sigma^{-1}(y - x + \Sigma b), (y - x + \Sigma b) \rangle \\ &= \frac{1}{2} \langle \Sigma^{-1}(y - x), (y - x) \rangle + \langle \Sigma^{-1}(y - x), \Sigma b \rangle + \frac{1}{2} \langle \Sigma^{-1} \Sigma b, \Sigma b \rangle \\ &= \frac{1}{2} \|y - x\|_\Sigma^2 + \langle y, b \rangle + \langle x, b \rangle + \frac{1}{2} \langle b, \Sigma b \rangle \end{aligned}$$

We rewrite the optimization problem as

$$\operatorname{prox}_{\Sigma F^*}(x) = \operatorname*{arg\,min}_{y \in \mathbb{R}^n} \frac{1}{2} \sum_{i=1}^n \frac{1}{\Sigma_i} (y_i - (x_i - \Sigma_i b_i))^2, \quad \text{s.t. } |y_i| \leq \alpha$$

Since the components of y are not coupled by the infinity norm in the constraint, the is a component-wise clipping of $x - \Sigma b$ to the interval $[-\alpha, \alpha]$ which shows the claim. □

Proposition B.13 (Prox Operator for Huber Loss Dataterm)**.** Let $b \in \mathbb{R}^n$, $\alpha, h > 0$ and let $\Sigma \in \mathbb{R}^{n \times n}$ be a positive definite diagonal matrix. Consider $F^* : \mathbb{R}^n \to \mathbb{R}$ with $F^*(p) = \langle p, b \rangle + \frac{h}{2\alpha} \|p\|_2^2 + \delta_{\|\cdot\|_\infty \leq \alpha}(p)$. Then the components of the weighted prox operator are given by

$$\left(\operatorname{prox}_{\Sigma F^*}(x)\right)_i = \frac{\alpha (x_i - \Sigma_i b_i)}{\max\{|x_i - \Sigma_i b_i|, \ h\Sigma_i + \alpha\}}$$

Proof. We re-write the optimization problem for the preconditioned prox operator

$$\begin{aligned}
&\operatorname{prox}_{\Sigma F^*}(x) = \underset{y\in\mathbb{R}^n}{\arg\min}\left\{\frac{1}{2}\|y-x\|_\Sigma^2 + \langle y, b\rangle + \frac{h}{2\alpha}\|y\|_2^2 + \delta_{\|\cdot\|_\infty\leq\alpha}(y)\right\} \\
\Leftrightarrow\quad &\operatorname{prox}_{\Sigma F^*}(x) = \underset{y\in\mathbb{R}^n}{\arg\min}\left\{\sum_{i=1}^n \frac{1}{2\Sigma_i}(y_i - x_i)^2 + y_i b_i + \frac{h}{2\alpha}y_i^2\right\}, \quad \text{s.t. } |y_i| \leq \alpha
\end{aligned}$$

We see that we can optimize the components separately. For each component the objective function is a parabola which obtains its unique minimum at

$$\begin{aligned}
0 &= \frac{1}{\Sigma_i}(\hat{y}_i - x_i) + b_i + \frac{h}{\alpha}\hat{y}_i \\
\Leftrightarrow\quad \hat{y}_i &= \frac{\alpha(x_i - \Sigma_i b_i)}{\alpha + h\Sigma_i}
\end{aligned}$$

To incorporate the constraints we need to distinguish the 3 cases shown below.

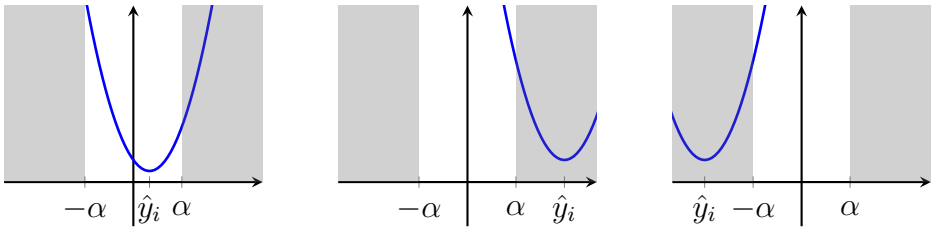


Figure 48: *Possible locations of the minimum $\hat{y}_i$ of the parabola relative to the interval $[-\alpha, \alpha]$*

If $\hat{y}_i$ is in the interval $[-\alpha, \alpha]$ we can accept it as the minimum. We can simplify the condition for this by

$$\begin{aligned}
&\left|\frac{\alpha(x_i - \Sigma_i b_i)}{\alpha + h\Sigma_i}\right| \leq \alpha \\
\Leftrightarrow\quad &|x_i - \Sigma_i b_i| \leq \alpha + h\Sigma_i
\end{aligned}$$

where we used that $\alpha, h, \Sigma_i > 0$. If $\hat{y}_i$ is not in the interval $[-\alpha, \alpha]$, the minimum is given by $\pm\alpha$ where the sign is chosen equal to the sign of $\hat{y}$, see Figure 48. So we can write the i-th component of the prox operator as

$$\begin{aligned}
\left(\operatorname{prox}_{\Sigma F^*}(x)\right)_i &= \begin{cases} \frac{\alpha(x_i - \Sigma_i b_i)}{\alpha + h\Sigma_i} & \text{if } |x_i - \Sigma_i b_i| \leq \alpha + h\Sigma_i \\ \operatorname{sgn}(x_i - \Sigma_i b_i)\,\alpha & \text{else} \end{cases} \\
&= \frac{\alpha(x_i - \Sigma_i b_i)}{\max\{|x_i - \Sigma_i b_i|,\ h\Sigma_i + \alpha\}}
\end{aligned}$$

where it is straightforward to verify the equivalence of the last line to the first formulation. □

A closed formulation of the preconditioned prox operator for the regularization terms is only possible under an additional assumption. In the next proposition, we state the prox operators and show in a subsequent remark that this condition is fulfilled in our case.

Proposition B.14 (Prox Operator for isotropic TV Regularization)**.** Let $\mathbf{\Gamma} \in \mathbb{R}^n_+$ and $\lambda_{reg} > 0$ and let $\Sigma \in \mathbb{R}^{2n\times 2n}$ be a positive definite diagonal matrix such that $\Sigma_i = \Sigma_{i+n},\ \forall i = 1, \dots n$. Consider $F^* : \mathbb{R}^{2n} \to \mathbb{R}$ with

$$F^*(p) = \delta_{\|\cdot\|_2 \leq \lambda_{reg}\Gamma_i}\left(p_{[i,i+n]}\right)$$

Then the i-th and the $i+n$-th component of the prox operator are given by

$$\left(\text{prox}_{\Sigma F^*}(x)\right)_{[i,i+n]} = \frac{\lambda_{reg}\Gamma_i}{\max\left\{\left\|x_{[i,i+n]}\right\|_2,\ \lambda_{reg}\Gamma_i\right\}} x_{[i,i+n]}$$

Proof. Using the addition condition on the matrix Σ and the short form $\gamma_i = \lambda_{reg}\Gamma_i$ we can write the formula for the preconditioned prox operator as

$$\begin{aligned}\text{prox}_{\Sigma F^*}(x) &= \underset{y\in\mathbb{R}^{2n}}{\arg\min}\left\{\frac{1}{2}\|y-x\|^2_\Sigma + F^*(y)\right\} = \\ &= \underset{y\in\mathbb{R}^{2n}}{\arg\min}\left\{\sum_{i=1}^{n}\frac{1}{2\Sigma_i}\left\|\begin{bmatrix}y_i\\ y_{i+n}\end{bmatrix} - \begin{bmatrix}x_i\\ x_{i+n}\end{bmatrix}\right\|_2^2\right\}, \quad \text{s.t. } \left\|y_{[i,i+n]}\right\|_2 \leq \gamma_i\end{aligned}$$

We see that we can optimize for the combination of the i-th and the $i+n$-th component separately. The result for these components is simply a projection of $x_{[i,i+n]}$ to the γ_i ball in the 2-norm. So we can write the i-th and the $i+n$-th component of the prox operator as

$$\begin{aligned}\left(\text{prox}_{\Sigma F^*}(x)\right)_{[i,i+n]} &= \begin{cases} x_{[i,i+n]} & \text{if } \left\|x_{[i,i+n]}\right\|_2 \leq \gamma_i \\ \frac{\gamma_i}{\left\|x_{[i,i+n]}\right\|_2} x_{[i,i+n]} & \text{else}\end{cases} \\ &= \frac{\gamma_i}{\max\left\{\left\|x_{[i,i+n]}\right\|_2,\ \gamma_i\right\}} x_{[i,i+n]}\end{aligned}$$

where the last equivalence is straightforward to verify.

□

Proposition B.15 (Prox Operator for isotropic Huber Regularization)**.** Let $\mathbf{\Gamma} \in \mathbb{R}^n_+$ and $\lambda_{reg}, h > 0$ and let $\Sigma \in \mathbb{R}^{2n\times 2n}$ be a positive definite diagonal matrix such that $\Sigma_i = \Sigma_{i+n},\ \forall i = 1, \dots n$. Consider $F^* : \mathbb{R}^{2n} \to \mathbb{R}$ with

$$F^*(p) = \sum_{i=1}^{n}\frac{h}{2\lambda_{reg}\Gamma_i}\left\|p_{[i,i+n]}\right\|_2^2 + \delta_{\|\cdot\|_2\leq\lambda_{reg}\Gamma_i}\left(p_{[i,i+n]}\right)$$

Then the i-th and the $i+n$-th component of the prox operator are given by

$$\left(\text{prox}_{\Sigma F^*}(x)\right)_{[i,i+n]} = \frac{\lambda_{reg}\Gamma_i}{\max\left\{\left\|x_{[i,i+n]}\right\|_2,\ \lambda_{reg}\Gamma_i + h\Sigma_i\right\}} x_{[i,i+n]}$$

Proof. Using the addition condition on the matrix Σ and the short form $\gamma_i = \lambda_{reg}\Gamma_i$ we can write the formula for the preconditioned prox operator as

$$\begin{aligned}&\text{prox}_{\Sigma F^*}(x) = \underset{y\in\mathbb{R}^{2n}}{\arg\min}\left\{\frac{1}{2}\|y-x\|^2_\Sigma + F^*(y)\right\} = \\ &= \underset{y\in\mathbb{R}^{2n}}{\arg\min}\left\{\sum_{i=1}^{n}\frac{1}{2\Sigma_i}\left\|\begin{bmatrix}y_i\\ y_{i+n}\end{bmatrix} - \begin{bmatrix}x_i\\ x_{i+n}\end{bmatrix}\right\|_2^2 + \frac{h}{2\gamma_i}\left\|y_{[i,i+n]}\right\|_2^2\right\}, \quad \text{s.t. } \left\|y_{[i,i+n]}\right\|_2 \leq \gamma_i\end{aligned}$$

We see that we can optimize for the combination of the i-th and the $i+n$-th component separately. The objective function for those two components is a quadratic form which obtains its unique (unconstrained) minimum $\hat{y}_{[i,i+n]}$ at

$$\begin{aligned} 0 &= \frac{1}{\Sigma_i}\left(\hat{y}_{[i,i+n]} - x_{[i,i+n]}\right) + \frac{h}{\gamma_i}\hat{y}_{[i,i+n]} \\ \Leftrightarrow \quad \hat{y}_{[i,i+n]} &= \frac{\gamma_i}{\gamma_i + h\Sigma_i} x_{[i,i+n]} \end{aligned} \tag{B.3}$$

The arguments to incorporate the constraints are similar to the ones in the proof of Proposition B.13, however, now in 2 dimensions. If the minimum of the quadratic form is located within the γ_i-ball around the origin, we can accept it. The corresponding condition can be simplified as

$$\begin{aligned} &\left\| \frac{\gamma_i}{\gamma_i + h\Sigma_i} x_{[i,i+n]} \right\|_2 \leq \gamma_i \\ \Leftrightarrow \quad &\left\| x_{[i,i+n]} \right\|_2 \leq \gamma_i + h\Sigma_i \end{aligned}$$

If the minimum $\hat{y}_{[i,i+n]}$ is located outside of the γ_i-ball, the constrained minimum is the projection of the unconstrained minimum to the γ_i-ball. To see this we decompose the argument of the objective function into $y_{[i,i+n]} = \hat{y}_{[i,i+n]} + \xi$ for $\xi \in \mathbb{R}^2$. We reformulate the objective function

$$\begin{aligned} f(\xi) &= \frac{1}{2\Sigma_i}\left\|\hat{y}_{[i,i+n]} + \xi - x_{[i,i+n]}\right\|_2^2 + \frac{h}{2\gamma_i}\left\|\hat{y}_{[i,i+n]} + \xi\right\|_2^2 \\ &= \frac{1}{2\Sigma_i}\left\| -\frac{h\Sigma_i}{\gamma_i + h\Sigma_i} x_{[i,i+n]} + \xi \right\|_2^2 + \frac{h}{2\gamma_i}\left\| \frac{\gamma_i}{\gamma_i + h\Sigma_i} x_{[i,i+n]} + \xi \right\|_2^2 \\ &= c_1 - \frac{h}{\gamma + h\Sigma_i}\langle x_{[i,i+n]}, \xi\rangle + \frac{1}{2\Sigma_i}\|\xi\|_2^2 + c_2 + \frac{h}{\gamma_i + h\Sigma_i}\langle x_{[i,i+n]}, \xi\rangle + \frac{h}{2\gamma_i}\|\xi\|_2^2 \\ &= \left(\frac{1}{2\Sigma_i} + \frac{h}{2\gamma_i}\right)\|\xi\|_2^2 + c_3 \end{aligned}$$

where c_1, c_2 and c_3 are constants that do not depend on ξ and we used the result of Equation (B.3) for the first step. So we see that the quadratic form is radial symmetric, i.e. the function value does only depend on the distance from the minimum and the value grows quadratically with increasing distance. However, this shows that the constrained minimum is indeed the projection of the unconstrained minimum to the γ_i-ball as the result is the (unique) feasible value that is closest to $\hat{y}_{[i,i+n]}$.

So we can write the i-th and the $i+n$-th component of the prox operator as

$$\begin{aligned} \left(\text{prox}_{\Sigma F^*}(x)\right)_{[i,i+n]} &= \begin{cases} \hat{y}_{[i,i+n]} & \text{if } \left\|x_{[i,i+n]}\right\|_2 \leq \gamma_i + h\Sigma_i \\ \frac{\gamma_i}{\left\|\hat{y}_{[i,i+n]}\right\|_2}\hat{y}_{[i,i+n]} & \text{else} \end{cases} \\ &= \begin{cases} \frac{\gamma_i}{\gamma_i + h\Sigma_i} x_{[i,i+n]} & \text{if } \left\|x_{[i,i+n]}\right\|_2 \leq \gamma_i + h\Sigma_i \\ \frac{\gamma_i}{\left\|x_{[i,i+n]}\right\|_2} x_{[i,i+n]} & \text{else} \end{cases} \\ &= \frac{\gamma_i}{\max\left\{\left\|x_{[i,i+n]}\right\|_2,\ \gamma_i + h\Sigma_i\right\}} x_{[i,i+n]} \end{aligned}$$

where again the last equivalence is straightforward to verify. □

Remark. It remains to check the additional condition $\Sigma_i = \Sigma_{i+n},\ \forall i = 1, \dots n$ for our case. We use the formula from to Theorem 4.4 for the diagonal entries of Σ, i.e.

$$\sigma_i = \frac{1}{\sum_{j=1}^{n} \left|\bar{D}_{ij}\right|^{\alpha}}$$

Note that since we have multiple dual variables we need to extract the respective rows from the general operator K. For the dual variable corresponding to the regularization this yields the extended discrete gradient operator $\bar{D}$.

First, we consider rows in $\bar{D}$ that correspond to the derivatives at inner pixels or pixels on the top or left boundaries. In those rows exactly 2 entries are 1 while all the others are zero. Therefore the corresponding entries σ_i are all equal to 2.

For pixels at the bottom or right boundary at least one of the corresponding rows in $\bar{D}$ is zero and therefore the respective entries in Σ are not the same. However, a zero row in K means that the respective component of the dual variable is not updated by $p^k + \Sigma K \bar{u}^k$ and therefore remains constant after the first step $p^{k+1} = \text{prox}_{\Sigma F^*}\left(p^k + \Sigma K \bar{u}^k\right)$. Moreover, due to the zero row the component of the dual variable has no influence on the update of the primal variable in $u^k - \Lambda K^T p^{k+1}$. Therefore, while not completely true, we can safely assume that $\Sigma_i = \Sigma_{i+n},\ \forall i = 1, \dots n$.

B.4 Gaussian Filtering

Gaussian filtering is a special form of linear filtering which is a discrete convolution of the image $I \in \mathbb{R}^{m \times n}$ with a filter kernel K. The general definition reads

$$I_{filt}[x, y] = \sum_{k,l} K[k, l]\, I[x - k, y - l] \tag{B.4}$$

In contrast to the convolution used for the bicubic interpolation we are considering the matrix representation of images only here. With our usual convention we denote $I[x, y] = I_{y,x}$. While K can be any function that is evaluated at integer points we often use *filter masks*, where we write K as a matrix. We then can intuitively understand the filtering as shifting the mask to the pixel $[x, y]$ and computing the weighted average of the neighboring pixels. We usually use symmetric filtering which requires the matrix to have odd dimensions. To keep the shifting interpretation consistent with Equation (B.4) we need to index the mask in the following way.

$K[1, 1]$	$K[0, 1]$	$K[-1, 1]$
$K[1, 0]$	$K[0, 0]$	$K[-1, 0]$
$K[1, -1]$	$K[0, -1]$	$K[-1, -1]$

(B.5)

Remark. Note that the indexing in Equation (B.5) is somewhat counter-intuitive as we would expect the element $K[-1, -1]$ in the top left corner. We could achieve this by replacing the minus signs in the image indexing in Equation (B.4) by plus signs, which fixes the indexing but instead contradicts the convolution definition. As filter masks are often given in this more obvious indexing, we have to rotate the matrix by 180° in this case to keep the shifting interpretation. For rotationally symmetric matrices (which are the most common class), this problem becomes obsolete.

For Gaussian filtering we use the Gaussian function

$$K[x,y] = ce^{-\frac{x^2+y^2}{2\sigma^2}} \tag{B.6}$$

where σ is the standard deviation and c is a generic normalizing constant. As this filter is separable we can write the filtering more efficiently by 2 convolutions with a 1D Gaussian kernel.

$$I_{filt}[x,y] = \sum_{k,l} K[k]\,K[l]\,I[x-k,y-l] \tag{B.7}$$

with

$$K[x] = ce^{-\frac{x^2}{2\sigma^2}} \tag{B.8}$$

Note that the kernel still has infinite support, and therefore all pixels are used for each pixel in the filtered image. However, since the kernel decays exponentially, we expect good approximations if we truncate the kernel. One way to design a mask that approximates a Gaussian kernel is to calculate the entries using Equation (B.8) directly. The constant c is used to scale the entries of the mask such that they sum to 1. This ensures that the average of the involved pixels stays constant over the filtering. The size of the mask is often coupled to the standard deviation σ of the Gaussian kernel. MATLAB's version of the filter (which is used in this thesis) uses a mask with $2 \cdot \lceil 2\sigma \rceil + 1$ entries.

A thorough discussion of linear and Gaussian filtering can be found e.g. in [22].

B.5 Gradient for the Isotropic Huber Regularization

We need to compute the gradient of the following function

$$\mathcal{R}(\mathbf{x}) = \sum_{i=1}^{mn} \Gamma_i \left\| x_{[i,i+mn]} \right\|_h \tag{B.9}$$

where $\|\cdot\|_h$ is the Huber norm with Huber parameter $h > 0$ according to Definition 3.2 and we used again the notation

$$x_{[i,i+n]} = \begin{bmatrix} x_i \\ x_{i+n} \end{bmatrix}$$

We repeat the definition of the Huber norm on for convenience.

$$\|x\|_h = \begin{cases} \frac{1}{2h}\|x\|_2^2 & \text{if } \|x\|_2 \le h \\ \|x\|_2 - \frac{h}{2} & \text{else} \end{cases}$$

It is immediate that we can calculate the i-th and the $i+mn$-th component of the gradient independently. Those components are given as

$$(\nabla \mathcal{R}(\mathbf{x}))_{[i,i+mn]} = \Gamma_i \begin{cases} \frac{1}{h} x_{[i,i+mn]} & \text{if } \left\| x_{[i,i+mn]} \right\|_2 \le h \\ \frac{1}{\left\| x_{[i,i+mn]} \right\|_2} x_{[i,i+mn]} & \text{else} \end{cases} \tag{B.10}$$

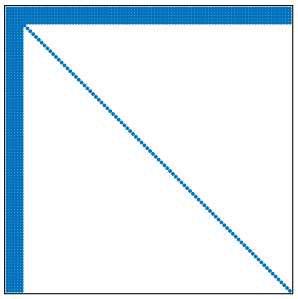


Figure 49: *Structure of the matrix Φ that we need to invert. The blue dots are non-zero entries while the rest of the entries is zero. The shown part is only an excerpt, the structure continues down and to the right.*

B.6 Schur Complement to Solve the Linear System

The inverted matrix Φ in Equation (5.36) has the structure shown in Figure 49. We can decompose it into the following block matrix

$$\Phi = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \tag{B.11}$$

where $A \in \mathbb{R}^{6|\mathcal{I}|\times 6|\mathcal{I}|}, B \in \mathbb{R}^{6|\mathcal{I}|\times mn}$ and $C \in \mathbb{R}^{mn\times 6|\mathcal{I}|}$ are full matrices and $D \in \mathbb{R}^{mn\times mn}$ is a diagonal matrix.

For this block matrix we now consider the following general linear system.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{B.12}$$

which we can write equivalently as

$$\begin{aligned} &\text{(I)} \quad Ax + By = f \\ &\text{(II)} \quad Cx + Dy = g \end{aligned} \tag{B.13}$$

We multiply equation (II) by $-BD^{-1}$ from the left and add it to equation (I). We simplify the resulting equation and compute x as

$$\begin{aligned} & Ax - BD^{-1}Cx = f - BD^{-1}g \\ \Leftrightarrow \quad & x = \left(A - BD^{-1}C\right)^{-1}\left(f - BD^{-1}g\right) \end{aligned} \tag{B.14}$$

Note that despite the high dimension we can easily invert the matrix D as it has a diagonal structure. The matrix $S = A - BD^{-1}C$ is called the Schur complement[8] and is likewise easy to invert as it has the same low dimension as A. Having obtained x we now can compute y in a second step.

$$y = D^{-1}\left(g - Cx\right) \tag{B.15}$$

where again D is easy to invert. An in depth discussion of the Schur Complement can be found e.g. in [48].

[8] Usually, the Schur complement of the matrix Φ is defined as $S = D - C^{-1}B$. This formula is equivalent to the definition given above in that we can transform one into the other by rearranging rows and columns. However, the form introduced above is better suited for our purpose.

References

[1] G. Aubert and P. Kornprobst. *Mathematical Problems in Image Processing*. Springer New York, 2006. doi:10.1007/978-0-387-44588-5.

[2] A. Beck. *First-Order Methods in Optimization*. Society for Industrial and Applied Mathematics, 2017. doi:10.1137/1.9781611974997.

[3] M. Born, E. Wolf, A. B. Bhatia, P. C. Clemmow, D. Gabor, A. R. Stokes, A. M. Taylor, P. A. Wayman, and W. L. Wilcock. *Principles of Optics*. Cambridge University Press, 1999. doi:10.1017/cbo9781139644181.

[4] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. doi:10.1017/cbo9780511804441.

[5] K. Bredies, K. Kunisch, and T. Pock. Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526, 2010. doi:10.1137/090769521.

[6] B. Chalmond. *Modeling and Inverse Problems in Imaging Analysis*. Springer New York, 2003. doi:10.1007/978-0-387-21662-1.

[7] A. Chambolle and T. Pock. A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145, 2010. doi:10.1007/s10851-010-0251-1.

[8] J. Civera, A. J. Davison, and J. M. M. Montiel. Inverse depth to depth conversion for monocular SLAM. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, apr 2007. doi:10.1109/robot.2007.363892.

[9] L. Condat. Discrete total variation: New definition and minimization. *SIAM Journal on Imaging Sciences*, 10(3):1258–1290, 2017. doi:10.1137/16m1075247.

[10] A. J. Davison. Real-time simultaneous localisation and mapping with a single camera. In *Proceedings Ninth IEEE International Conference on Computer Vision*. IEEE, 2003. doi:10.1109/iccv.2003.1238654.

[11] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(6):1052–1067, 2007. doi:10.1109/tpami.2007.1049.

[12] F. Devernay and O. D. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. In *Investigative and Trial Image Processing*, volume 2567. SPIE, 1995. doi:10.1117/12.218487.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. URL: http://carla.org/.

[14] D. Drusvyatskiy and A. S. Lewis. Error bounds, quadratic growth, and linear convergence of proximal methods. *Mathematics of Operations Research*, 43(3):919–948, 2018. doi:10.1287/moor.2017.0889.

[15] D. Drusvyatskiy and C. Paquette. Efficiency of minimizing compositions of convex functions and smooth maps. *Mathematical Programming*, 2018. doi:10.1007/s10107-018-1311-3.

[16] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, 2018. doi:10.1109/tpami.2017.2658577.

[17] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Computer Vision – ECCV 2014*, pages 834–849. Springer International Publishing, 2014. doi:10.1007/978-3-319-10605-2_54.

[18] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In *2013 IEEE International Conference on Computer Vision*. IEEE, dec 2013. doi:10.1109/iccv.2013.183.

[19] D. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Always learning. Pearson, 2012. URL: https://www.pearson.com/us/higher-education/program/Forsyth-Computer-Vision-A-Modern-Approach-2nd-Edition/PGM111082.html.

[20] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. doi:10.1017/cbo9780511811685.

[21] C. Hertzberg. A framework for sparse, non-linear least squares problems on manifolds, 2008. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.174.7485.

[22] R. Jain, R. Kasturi, and B. G. Schunck. *Machine Vision*. McGraw-Hill, Inc., New York, NY, USA, 1995.

[23] R. Keys. Cubic convolution interpolation for digital image processing. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 29(6):1153–1160, 1981. doi:10.1109/tassp.1981.1163711.

[24] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2007. doi:10.1109/ismar.2007.4538852.

[25] G. Kuschk. *Efficient Large-Scale Stereo Reconstruction using Variational Methods*. Dissertation, Technische Universität München, München, 2019. URL: https://mediatum.ub.tum.de/1455869.

[26] J. M. Lee. *Introduction to Smooth Manifolds*. Springer New York, 2012. doi:10.1007/978-1-4419-9982-5.

[27] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944. doi:10.1090/qam/10666.

[28] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry. *An Invitation to 3-D Vision*. Springer New York, 2004. doi:10.1007/978-0-387-21779-6.

[29] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963. doi:10.1137/0111030.

[30] S. Martull, M. Peris, and K. Fukui. Realistic cg stereo image dataset with ground truth disparity maps. In *ICPR workshop TrakMark2012*, 2012. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.454.3938.

[31] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. doi:10.1109/tro.2015.2463671.

[32] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision*. IEEE, 2011. doi:10.1109/iccv.2011.6126513.

[33] P. Ochs, J. Fadili, and T. Brox. Non-smooth non-convex bregman minimization: Unification and new algorithms. *Journal of Optimization Theory and Applications*, 181(1):244–278, dec 2018. doi:10.1007/s10957-018-01452-0.

[34] M. Peris, S. Martull, A. Maki, Y. Ohkawa, and K. Fukui. Towards a simulation driven stereo vision system. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 1038–1042. IEEE, 2012. URL: https://ieeexplore.ieee.org/abstract/document/6460313.

[35] T. Pock and A. Chambolle. Diagonal preconditioning for first order primal-dual algorithms in convex optimization. In *2011 International Conference on Computer Vision*. IEEE, 2011. doi:10.1109/iccv.2011.6126441.

[36] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. An algorithm for minimizing the mumford-shah functional. In *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009. doi:10.1109/iccv.2009.5459348.

[37] T. Pock, L. Zebedin, and H. Bischof. TGV-fusion. In *Lecture Notes in Computer Science*, pages 245–258. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-19391-0_18.

[38] M. Quan, S. Piao, M. Tan, and S.-S. Huang. Accurate monocular visual-inertial SLAM using a map-assisted EKF approach. *IEEE Access*, 7:34289–34300, 2019. doi:10.1109/access.2019.2904512.

[39] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun. Dense monocular depth estimation in complex dynamic scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2016. doi:10.1109/cvpr.2016.440.

[40] R. T. Rockafellar. *Convex analysis*. Princeton university press, 1970. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.421.55.

[41] L. I. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992. doi:10.1016/0167-2789(92)90242-f.

[42] J. Stühmer, S. Gumhold, and D. Cremers. Real-time dense geometry from a handheld camera. In *Lecture Notes in Computer Science*, pages 11–20. Springer Berlin Heidelberg, 2010. doi:10.1007/978-3-642-15986-2_2.

[43] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment — a modern synthesis. In *Vision Algorithms: Theory and Practice*, pages 298–372. Springer Berlin Heidelberg, 2000. doi:10.1007/3-540-44480-7_21.

[44] L. Valgaerts, A. Bruhn, M. Mainberger, and J. Weickert. Dense versus sparse approaches for estimating the fundamental matrix. *International Journal of Computer Vision*, 96(2):212–234, 2011. doi:10.1007/s11263-011-0466-7.

[45] A. Wedel, T. Pock, C. Zach, H. Bischof, and D. Cremers. An improved algorithm for tv-l1 optical flow. In *Lecture Notes in Computer Science*, pages 23–45. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-03061-1_2.

[46] M. Werlberger, W. Trobin, T. Pock, A. Wedel, D. Cremers, and H. Bischof. Anisotropic huber-l1 optical flow. In *Procedings of the British Machine Vision Conference 2009*. British Machine Vision Association, 2009. doi:10.5244/c.23.108.

[47] C. Zach, T. Pock, and H. Bischof. A globally optimal algorithm for robust tv-l^1 range image integration. In *2007 IEEE 11th International Conference on Computer Vision*, pages 1–8. IEEE, 2007. doi:10.1109/ICCV.2007.4408983.

[48] F. Zhang, editor. *The Schur Complement and Its Applications*. Springer-Verlag, 2005. doi:10.1007/b105056.