
Variational L_0 Regularization for Enhanced Depth Image Analysis

DISSERTATION

zur Erlangung des Grades eines
Doktors der Ingenieurwissenschaften (Dr.-Ing.)

vorgelegt von
ANDREAS GÖRLITZ, M.Sc.

eingereicht bei der Naturwissenschaftlich-Technischen Fakultät
der Universität Siegen

Siegen 2025

Betreuer und erster Gutachter

PROF. DR. ANDREAS KOLB

Universität Siegen

Zweiter Gutachter

PROF. DR. MICHAEL MÖLLER

Universität Siegen

Tag der mündlichen Prüfung

10. Oktober 2025

*Für jene, die früh gehen mussten, und jene, die mich lange begleitet haben.
Тем, кто ушёл слишком рано, и тем, кто долго был рядом.*

Vera Petrova (23.05.1948 – 07.02.1994)
Вера Петрова

Wilhelm Görlitz (07.05.1938 – 03.04.2003)
Василий Герлиц

Frida Görlitz (08.09.1942 – 01.02.2025)
Фрида Герлиц

DANKSAGUNG

Zuallererst möchte ich Prof. Dr. Andreas Kolb meinen tiefsten Dank aussprechen – für die Chance, in einem interdisziplinären und interkulturellen Team mit so vielen brillanten Köpfen zusammenzuarbeiten. Besonders dankbar bin ich ihm für seine großartige Unterstützung während der gesamten Zeit, sei es durch fachliche Impulse, konstruktives Feedback oder seine unermüdliche Hilfsbereitschaft – selbst bis spät in die Nacht vor wichtigen Paper-Deadlines. Seine kreativen Ideen, die oft mit den Worten "Man könnte ja auf die dumme Idee kommen [...]" beginnen, haben jede einzelne Publikation bereichert.

Ebenso gilt mein aufrichtiger Dank Prof. Dr. Michael Möller für seine wertvolle und ebenfalls unermüdliche Unterstützung in entscheidenden Phasen, sowie für sein kritisches Feedback, das maßgeblich zur kontinuierlichen Verbesserung meiner Arbeit beigetragen hat.

Ich danke Jonas Geiping herzlich für seine Unterstützung bei der Einarbeitung in das Thema Variational Optimization, das die Grundlage dieser Arbeit bildet und in einer gemeinsamen Publikation mündete.

Mein Dank gilt auch allen Kolleginnen und Kollegen der Computer Graphics- und der Computer-Vision Gruppe für ihre Unterstützung, ihr Feedback, bereichernde Gespräche und gemeinsame Unternehmungen, insbesondere Markus Kluge, Hendrik Hochstetter, Dmitri Presnov, Hendrik Sommerhoff, Hartmut Bauermeister, Ulrich Schipper, Willi Gräfrath und Sarah Wagener.

Mein herzlicher Dank gilt meinen Schwiegereltern, Claudia und Karl Seeger, u.a. für die unschätzbare Unterstützung bei der Kinderbetreuung. Dank Eurem unermüdlichen Einsatz – von unzähligen LEGO-Bauten bis hin zu zahlreichen Besuchen in Zoos und Museen – konnte ich wertvolle Zeit für das Verfassen dieser Arbeit gewinnen.

Ich danke herzlichst meinen Eltern, Swetlana und Alexander Görlitz, für die bedingungslose Unterstützung seit meiner Kindheit. Ihr seid mit nichts als einem Koffer und einen kleinen Jungen an der Hand in ein fremdes Land gekommen, und habt durch viel Fleiß und Hingabe die Grundlage erschaffen für das Leben, das ich heute haben darf. Auch Ihr verwöhnt Eure Enkelkinder mit viel Liebe – und noch mehr Süßigkeiten – und habt mir so viel Zeit zum Schreiben geschenkt.

Zum Abschluss gilt mein tiefster Dank meiner wunderbaren Familie – meiner Frau Annabell und unseren beiden Kindern Anton Aeneas und Linnea Aurelia. Ich kann mit Worten nicht ausdrücken, wie stolz und glücklich ihr mich macht. Ihr holt mich an die frische Luft, wenn ich mich zu sehr in der Arbeit verliere und zaubert stets ein Lächeln auf mein Gesicht. Ich liebe euch von ganzem Herzen!

ABSTRACT

Mathematical optimization is fundamental in computational imaging, where problems often center on minimizing an objective function under various constraints. An important class of optimization problems involves minimal partitioning, which aims to segment data into partitions which meet certain criteria, such as photometric or geometric similarity. This class of problems is computationally demanding as it has a $\mathcal{NP}$ -Hard complexity, which means that the computational effort required to estimate a solution increases exponentially with the size of the data. Consequently, finding (nearly) optimal solutions within a reasonable amount of time is often infeasible.

This dissertation presents an analysis of the efficient and effective utilization of minimal partitioning techniques in RGB-D image processing. This is achieved by constraining the underlying optimization problems with the L_0 “norm”, which enforces piecewise constant solutions.

The first case study analyses the utilization of the L_0 “norm” for scene flow estimation, where 3D motions are modeled as rigid transformations and constrained to be piecewise constant. By substituting the L_0 “norm” with the *truncated quadratic* “norm”, this approach achieves both high accuracy as well as real-time performance.

The next focus is on the estimation of geometric models on RGB-D images, specifically to obtain denoised depth and normal values. The novel geometric model definition along with the *Cut Pursuit* optimization leads to a higher accuracy compared to other model based methods.

The third and final enhancement addresses the high computational demands of the preceding *Cut Pursuit* based method by introducing a stochastic optimization technique. It improves the convergence rate and also the overall runtime efficiency.

ZUSAMMENFASSUNG

Mathematische Optimierung bildet die Basis in der computergestützten Bildgebung, bei der Probleme als Minimierung einer Zielfunktion unter verschiedenen Nebenbedingungen formuliert werden.

Eine wichtige Klasse von Optimierungsproblemen umfasst die sogenannte *minimale Partitionierung*, die darauf abzielt, Daten in Partitionen zu segmentieren, die bestimmte Kriterien erfüllen, wie z. B. photometrische oder geometrische Ähnlichkeit. Das Lösen von Problemen dieser Klasse ist mit hoher Rechenlast verbunden, da sie $\mathcal{NP}$ -Schwere Komplexität aufweist. Das bedeutet, dass der Rechenaufwand für die Lösungsschätzung exponentiell mit der eingegebenen Datenmenge wächst. Daher ist es i. d. R. nicht möglich (nahezu) optimale Lösungen in hinreichend kurzer Zeit zu berechnen.

Diese Dissertation stellt eine Analyse der effizienten und effektiven Nutzung von minimalen Partitionsproblemen in der RGB-D-Bildverarbeitung vor. Dies wird hauptsächlich durch die Beschränkung der zugrunde liegenden Optimierungsprobleme mit der L_0 "norm" erreicht, welche stückweise konstante Lösungen erzwingt.

Die erste Fallstudie dieser Arbeit analysiert die Nutzung der L_0 "norm" für die Schätzungen des sog. *scene flows*. Hierbei werden 3D-Bewegungen als rigide Transformationen modelliert und als stückweise konstant angenommen. Durch das Ersetzen der L_0 "norm" mit der *truncated quadratic* "norm" wird sowohl eine hohe Genauigkeit als auch Echtzeit-Performanz erreicht.

Der nächste Schwerpunkt liegt auf der Schätzung geometrischer Modelle auf RGB-D Bildern, insbesondere zur Ermittlung von rauschreduzierten Tiefen- und Normalenwerten. Die neuartige Definition des geometrischen Modells zusammen mit dem *Cut Pursuit*-Verfahren führt zu höherer Genauigkeit im Vergleich zu anderen modellbasierten Methoden.

Die dritte und letzte Verbesserung reduziert die Rechenanforderungen des vorherigen *Cut Pursuit* Verfahrens durch die Nutzung eines stochastischen Optimierungsansatzes. Dies verbessert die Konvergenzrate und steigert die Gesamteffizienz der Laufzeit des Verfahrens.

CONTENTS

1	Introduction	1
1.1	Challenges	2
1.2	Contributions	3
1.2.1	L_0 “norm” Regularization for Scene Flow Estimation on RGB-D images	3
1.2.2	Parameter Estimation for Non-linear Models in RGB-D Image Denoising	3
1.2.3	Fast L_0 Partitioning with Stochastic Proximal Gradient Descent	4
1.2.4	Publications	4
1.3	Outline	5
2	Fundamentals	7
2.1	Depth Cameras	7
2.1.1	Pinhole Camera Model	8
2.1.2	Camera Calibration	11
2.1.3	Point and Normal Maps	11
2.2	Mathematical Fundamentals	12
2.2.1	Norms	12
2.2.2	Definitions from Set Theory	13
2.2.3	Definitions from Functional Analysis	15
2.2.4	Indicator Function	17
2.2.5	Support Function	17
2.3	Finite Weighted Graphs	19
2.4	Variational Optimization	22
2.4.1	Convex Conjugate	22
2.4.2	Proximal Mapping	24
2.4.3	Primal Dual Hybrid Gradient (PDHG)	28
2.4.4	Diagonally Preconditioned PDHG	29
2.4.5	PDHG on Finite Weighted Graphs	30
3	Image Partitioning on Finite Weighted Graphs	33
3.1	Partitioning via Truncated Quadratics	34
3.1.1	Derivation	34
3.1.2	Results	37
3.2	Partitioning via Cut Pursuit	38
3.2.1	Derivation	38
3.2.2	Results	40
4	Piecewise Rigid Scene Flow Estimation	43
4.1	Motivation	43

4.2	Related Work	45
4.3	Proposed Method	47
4.3.1	Scene Flow	47
4.3.2	Rigid Scene Flow	47
4.3.3	Linearized Rigid Scene Flow	49
4.3.4	Optimization Problem Formulation	49
4.3.5	Optimization Problem Linearization	51
4.3.6	Primal-Dual Formulation on Finite Weighted Graphs	53
4.4	Evaluation	55
4.4.1	Evaluation without Reference Flows	56
4.4.2	Evaluation with Reference Flows	57
4.5	Conclusion	58
5	Coherent Denoising of Depth Images and Normal Maps	59
5.1	Motivation	59
5.2	Related Work	61
5.3	Proposed Method	63
5.3.1	Inverse Depth Map	64
5.3.2	Second-order Geometry Model	64
5.3.3	Coherent Depth Image and Normal Map	65
5.4	Evaluation	69
5.4.1	Evaluation on Clean ICL NUIM Dataset	70
5.4.2	Evaluation on Noisy ICL NUIM Dataset	71
5.4.3	Evaluation on Real-World NYUv2 Dataset	73
5.5	Conclusion	74
6	FL0C: Fast L_0 Cut Pursuit	75
6.1	Motivation	75
6.2	Related Work	76
6.3	Proposed Method	77
6.3.1	Fast Cut Pursuit	78
6.3.2	Fast Binary Partitioning	79
6.3.3	PDHG Solver	80
6.3.4	Stochastic Solver	81
6.4	Evaluation	83
6.5	Conclusion	85
7	Conclusion	87

NOTATIONS

Acronyms and Abbreviations

SL	Structured Light
ToF	Time-of-Flight
RGB	Red-Green-Blue
RGB-D	RGB and Depth
ROF	Rudin Osher Fatemi
MS	Mumford-Shah
PCMS	Piecewise constant Mumford Shah
PDHG	Primal-dual hybrid gradient
l.s.c.	lower semi continuous

Set Theory

$\mathbb{N}, \mathbb{R}$	Sets of natural and real numbers
$\mathbb{X}$	Arbitrary set
$\mathbb{K}$	Arbitrary convex set
$\mathbb{X}^*$	Dual set of $\mathbb{X}$
$\partial\mathbb{X}$	Boundary of set $\mathbb{X}$
$\text{Per}(\mathbb{X})$	Perimeter of set $\mathbb{X}$
$\delta_{\mathbb{X}}(\cdot)$	Indicator function of set $\mathbb{X}$
$\sigma_{\mathbb{X}}(\cdot)$	Support function of set $\mathbb{X}$
$\mathcal{H}_i(\mathbf{b})$	Supporting hyperplane at boundary point $\mathbf{b} \in \partial\mathbb{X}$

Linear Algebra

s	Scalar variable $s \in \mathbb{R}$
$\mathbf{v}$	Vector valued variable $\mathbf{v} \in \mathbb{R}^n$
K	Matrix $K \in \mathbb{R}^{n \times m}$
$\ \cdot\ _p$	L_p norm
$\ \cdot\ _p^*$	Dual norm of L_p norm
$\ \cdot\ _0$	L_0 "norm"
$\mathbb{B}_p$	Unit ball of L_p norm

Analysis

$f : \mathbb{X} \rightarrow \mathbb{R}$	Arbitrary function f from $\mathbb{X}$ to $\mathbb{R}$
$\text{dom}(f)$	Domain of the function f
$\text{gra}(f)$	Graph of the function f
$\text{epi}(f)$	Epigraph of the function f
$\partial f(\mathbf{x})$	Subdifferential of the function f at $\mathbf{x}$
∇f	Gradient of the function f

Optimization

G, F	G and F terms of PDHG algorithm
ϕ_D, ϕ_R	Data and regularization terms
f^*, f^{**}	Conjugate and biconjugate of f
$M_{\lambda f}$	Moreau envelope
$\text{prox}_{\tau f}(\cdot)$	Proximal operator
Γ_0	Set of proper, convex and l.s.c. functions

Images

C	camera matrix
f_x, f_y, c_x, c_y	Intrinsic camera parameters
Ω	Image domain
$[p_x, p_x] \in \Omega$	Pixel coordinate
$\mathcal{I}$	Color image
$\mathcal{D}$	Depth image
$\mathcal{V}$	Point map
$\mathcal{N}$	Normal map

LIST OF FIGURES

Figure 2.1	Pinhole camera model in 2D.	8
Figure 2.2	Pinhole camera model in 3D.	9
Figure 2.3	Boundary, interior, boundary point and convex hull of a set. . .	14
Figure 2.4	Supporting hyperplanes of a set.	15
Figure 2.5	Graph, domain, and epigraph of a function.	16
Figure 2.6	Support function of a convex set.	18
Figure 2.7	Finite weighted graph.	20
Figure 2.8	Relation between convex conjugate and biconjugate of a function.	23
Figure 2.9	Moreau envelope of a function.	26
Figure 2.10	Projections to unit balls $\mathbb{B}_1$, $\mathbb{B}_2$ and $\mathbb{B}_\infty$	27
Figure 3.1	Results of partitioning via truncated quadratics.	37
Figure 3.2	Sketch of <i>Cut Pursuit</i> partitioning.	39
Figure 3.3	Results of partitioning via Cut Pursuit quadratics.	41
Figure 4.1	Effect of 3D scene flow on 2D image coordinates.	48
Figure 4.2	Successive linearizations and warpings.	52
Figure 4.3	Color-coded projections of scene flow estimates.	56
Figure 4.4	Scene flow on static scenes.	57
Figure 4.5	Visual comparison of motion segments for VOSF and $\mathbf{L}_0$ approach.	58
Figure 5.1	Colored point cloud with surface normals.	60
Figure 5.2	Workflow of the Cut Pursuit algorithm.	67
Figure 5.3	Illustration of a merge step.	68
Figure 5.4	Results on clean ICL NUIM dataset	70
Figure 5.5	Results on noisy ICL NUIM dataset.	71
Figure 5.6	Result (shown as point cloud) of particular frame from the noisy ICL NUIM dataset.	72
Figure 5.7	Results on NYUv2 dataset.	73
Figure 5.8	Result (shown as point cloud) of one particular frame from the NYUv2 dataset.	73
Figure 6.1	Illustration of a sparse finite weighted graph.	77
Figure 6.2	Procedure of Cut Pursuit on a given graph with two partitions. .	78
Figure 6.3	Quantitative comparison of variational graph cut results.	84
Figure 6.4	Results after five Cut Pursuit iterations.	85

LIST OF TABLES

Table 4.1	RGB-D image pairs used for evaluation.	55
Table 4.2	PE and GE metrics for various datasets.	56
Table 4.3	Photometric errors (PE), geometric errors (GE) and end-point errors (EPE) for the datasets shown in Fig. 4.4.	57
Table 5.1	Programming languages and runtimes.	70
Table 6.1	Runtime performance of SPDHG for different batch sizes B	83

LIST OF ALGORITHMS

2.1	Primal Dual Hybrid Gradient (PDHG) Algorithm "Alg1"	28
2.2	Primal Dual Hybrid Gradient (PDHG) Algorithm "Alg2"	29
2.3	PDHG Alg1 on Finite Weighted Graphs	31
2.4	PDHG Alg2 on Finite Weighted Graphs	32
3.1	Cut Pursuit Algorithm	40
4.1	L_0 Rigid Scene Flow Estimation	54
6.1	PDHG "Alg1" Solver of Equation (6.11)	81
6.2	Proposed Stochastic PDHG (SPDHG) Algorithm	82

INTRODUCTION

Minimal partitioning problems (MPP) play a central role in various applications in computer vision and image processing domains. The goal is to divide an image into distinct regions with similar characteristics, while minimizing the boundary length of each partition. The underlying optimization formula builds the foundation of various algorithms, including visual odometry, simultaneous localization and mapping (SLAM), scene flow estimation, and 3D reconstruction of both static and dynamic scenes [Zol+18].

Minimal partitioning

A main property of minimal partitioning is sparsity, which means that changes mainly occur at the boundaries of partitions. The L_0 “norm” is especially relevant in this context, as it offers a direct way to quantify sparsity by counting the number of non-zero elements. Consequently, the solution to an optimization problem which includes L_0 “norm” based constraints will have fewest non-zero elements, making it as sparse as possible.

Sparsity and L_0 “norm”

Sparse data can be reduced by storing only the non-zero elements of each partition. This compression allows for more efficient processing of large datasets or computationally demanding algorithms. As demonstrated by Geiping et al. [Gei+20], such data reduction can speed up computationally intensive tasks, such as lifting, while preserving the result of the original dense data.

Data reduction

Over the past decade, the availability of low-cost depth cameras, such as Microsoft Kinect and Intel RealSense, has significantly influenced computer vision. Many existing algorithms for color images were adapted and applied on depth images, e.g. to reduce noise. However, depth data introduces additional challenges and requires specialized processing to maximize performance. Thus many approaches apply sparsity constraints at later stages, typically on secondary structures such as meshes [NB15] or point clouds [SSW15; TGB19], which can reduce the potential benefits of sparsity at earlier stages.

Depth cameras

This thesis addresses these limitations by introducing methods that incorporate L_0 “norm” in the initial stages of processing RGB-D (color and depth) images. By applying the L_0 “norm” early on RGB-D data, this work aims to improve algorithmic efficiency and optimize performance in computer vision tasks which include RGB-D data.

1.1 CHALLENGES

MPP is $\mathcal{NP}$ -Hard

Minimal partitioning problems and the optimization of the L_0 “norm” are very challenging tasks due to their $\mathcal{NP}$ -Hard complexity. To bypass the computational impediments involved in solving these kind of problems, the L_1 norm is often used as a simpler alternative, which can be seen as a convex relaxation of the L_0 “norm”. This substitution was pioneered in the context of image denoising by Rudin, Osher, and Fatemi (ROF) [ROF92], who utilized the L_1 norm as the regularization and the squared L_2 norm for the data term. The squared L_2 norm is most effective when dealing with zero-mean Gaussian noise in observations. For other types of noise the data term must be chosen appropriately, e.g. if the data is perturbed by outliers then the L_1 norm should be utilized instead [CP16].

Several proposals emerged in attempt to enhance the approximation of the L_0 “norm”, i.e. to increase the sparsity, compared to the L_1 norm. Primarily originating from the *sparse coding* domain, alternatives such as the L_q pseudo-norm ($0 < q < 1$), the L_1/L_2 norm, and the L_1 - L_2 norm [Lou+15], along with the truncated quadratic “norm” [BVZ01; SC14] from the computer vision and image processing domains, have been introduced as promising substitutes to the L_0 “norm”.

However, using the L_0 “norm” or any alternatives on range data brings its own challenges, stemming from the inherent characteristics of depth images. For example, in contrast to color images, the gradient of depth images is not sparse. Consequently, optimization techniques applied to the gradient tend to have a strong fronto-parallel bias toward the observer’s viewpoint. This bias can be mitigated by employing computationally demanding high-order gradient regularization techniques, such as the Total Generalized Variation (TGV) [BKP10; Fer+13].

Another notable difference lies in the distortions of depth images, which in contrast to color images may not only include highly inaccurate but also missing values. The magnitude of these disturbances varies depending on parameters like the distance between the depth camera and the object or the pixel coordinates, as analyzed by Sarbolandi et al. [SLK15]. This noise is further amplified in the estimation of surface normals, which are typically computed based on the first gradient of the depth image. This issue is often tackled using basic denoising methods like the bilateral filter [TM98], which are applied to depth images and normal maps separately. However, such simple techniques may introduce discrepancies between normal vectors and depth values, leading to incoherent geometric information.

Model-based methods that estimate planar surfaces [FTK14; RMB21] or cylinders [PG18] have shown promise in providing accurate depth estimates on noisy data without a fronto-parallel bias. Moreover, these models can be used to yield the respective surface normals, leading to coherent depth and normal values. Nonetheless, these methods are constrained by the underlying geometric model and often exhibit poor performance in scenes with complex geometries.

Non-sparse gradients

Incoherent depth and normals

Simple models

1.2 CONTRIBUTIONS

This dissertation gives a detailed analysis of the applicability of *minimal partitioning problems* within the realms of RGB-D image processing, focusing on the effective and efficient utilization of the L_0 “norm”. The following contributions have been made in the scope of this work.

1.2.1 L_0 “norm” Regularization for Scene Flow Estimation on RGB-D images

First the potential of the L_0 “norm”, in the area of scene flow estimation on RGB-D images, is examined. The primary contribution lies in representing 3D motions as piecewise constant rigid transformations, alongside an analysis of the L_0 “norm” to estimate these transformations. The rotational part of each transformation is assumed to be small and is thus represented as a linear operator. This trick helps to simplify the derived formulas and to reduce the optimization complexity. The computational hurdles inherent in solving $\mathcal{NP}$ -Hard problems, are bypassed by using the *truncated quadratic “norm”* as a substitute for the L_0 “norm”. This way this approach achieves both high accuracy as well as real-time performance, compared to state-of-the-art methods. Nonetheless, there are slight variations in the results due to numerical reasons, which primarily stem from the use of the *truncated quadratic “norm”*. Thus overall the calculated results are not sparse. This issue is addressed, by performing a lightweight post-processing step, namely value binning, to enhance sparsity.

1.2.2 Parameter Estimation for Non-linear Models in RGB-D Image Denoising

The next main contribution is a direct approach of computing sparse results, without additional post-processing steps. To achieve this in the context of RGB-D image denoising, the respective RGB-D image is divided into spatially coherent partitions, and a non-linear geometric model is estimated per partition. Subsequently all points of a partition which do not fit to the estimated model, are divided via *graph cuts* into new partitions. This process is repeated until convergence, i.e. until all points of a RGB-D image fit to the respective geometric model. It is shown that the non-linear geometric model is equal to a quadric surface, such that the surface normals can be derived from the model parameters. This allows to denoise depth and normal values, such that they are coherent to each other. Compared to other model based methods, the presented approach shows a better accuracy in denoising both, depth and normal values. Furthermore the estimated model parameters are constant per partition, thus the gradient of these model parameters is sparse. However, due to the extensive use of the underlying *graph cut* algorithm, this approach is computationally demanding, compared to the previous method based on the *truncated quadratic “norm”*.

1.2.3 Fast L_0 Partitioning with Stochastic Proximal Gradient Descent

Finally, to improve the runtime performance of the previous approach, a stochastic optimization method is utilized in this thesis. This contribution leads to smaller computational demands of the overall algorithm, since only a subset of all pixels of the RGB-D image are updated during the optimization process. As the analysis reveals, the convergence rate of this approach is improved significantly by using this stochastic optimization method. However, since this approach accesses memory in a random instead of a linear way, this approach is not *per-se* faster in terms of runtime despite its better convergence rate. This issue is tackled by using an enhanced memory access pattern strategy, which helps to improve the runtime performance.

1.2.4 Publications

The following is a list of all publications achieved during this research. The contributions made in this thesis have been presented in [GGK19], [GMK22], and [GMK24]. The basis for RGB-D algorithms emerged from research on 3D reconstruction with RGB-D cameras [Zol+18].

- [Zol+18] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. “State of the Art on 3D Reconstruction with RGB-D Cameras.” In: *Computer Graphics Forum* 37.2 (2018), pp. 625–652. DOI: [10.1111/cgf.13386](https://doi.org/10.1111/cgf.13386)
- [GGK19] A. Görlitz, J. Geiping, and A. Kolb. “Piecewise Rigid Scene Flow with Implicit Motion Segmentation.” In: *IEEE International Conference on Intelligent Robots and Systems*. 2019, pp. 1758–1765. DOI: [10.1109/IR0540897.2019.8968018](https://doi.org/10.1109/IR0540897.2019.8968018)
- [GMK22] A. Görlitz, M. Möller, and A. Kolb. “FL0C: Fast L_0 Cut Pursuit for Estimation of Piecewise Constant Functions.” In: *IEEE International Conference on Image Processing*. 2022, pp. 3677–3681. DOI: [10.1109/ICIP46576.2022.9897892](https://doi.org/10.1109/ICIP46576.2022.9897892)
- [GMK24] A. Görlitz, M. Möller, and A. Kolb. “Coherent Enhancement of Depth Images and Normal Maps Using Second-Order Geometric Models on Weighted Finite Graphs.” In: *International Conference on 3D Vision*. 2024, pp. 623–630. DOI: [10.1109/3DV62453.2024.00023](https://doi.org/10.1109/3DV62453.2024.00023)

1.3 OUTLINE

The remainder of this thesis is structured as follows.

Chapter 2 provides the essential theoretical background for the chapters that follow. It begins with a brief overview of RGB-D cameras and the pinhole camera model, followed by an in-depth explanation of mathematical foundations for convex optimization. The focus is on variational optimization, which covers convex conjugates, the proximal operator, and the primal dual hybrid gradient (PDHG) method.

Chapter 3 discusses image partitioning within the context of finite weighted graphs. It presents a solution based on *truncated quadratics*, which will be applied in chapter 4. Additionally, it introduces the *Cut Pursuit* method for solving this problem, which will be utilized in chapters 5 and 6.

Chapter 4 describes how to leverage the L_0 “norm” to enhance the robustness of algorithms in the context of RGB-D images. This chapter uses scene flow estimation as a case study to demonstrate the advantages of the L_0 “norm”. It illustrates how this norm can effectively address underdetermined optimization problems on noisy RGB-D input images.

Chapter 5 presents a model based approach for denoising of depth images through the L_0 “norm”. It partitions the depth image in regions of similar high-order geometric model parameters, which then allow to compute corresponding depth and normal values. This results in a unified and coherent representation of depth and normal information.

Chapter 6 introduces the acceleration of the optimization approach from the previous chapter. This is achieved through the implementation of a stochastic primal-dual hybrid gradient method (SPDHG) and a convex formulation of the problem, which enhances the convergence rate as well as the overall runtime performance.

Chapter 7 concludes this thesis and proposes possible directions of future work.

FUNDAMENTALS

This chapter lays the foundation for the remainder of this thesis. It introduces key terminology related to depth cameras and explains the pinhole camera model. Furthermore, it provides an overview of finite weighted graphs, convex optimization, and the Primal-Dual Hybrid Gradient (PDHG) solver proposed by Chambolle and Pock (2011) [CP11].

2.1 DEPTH CAMERAS

Depth cameras are devices capable of capturing and providing distance information in the observed scene. They generate so-called *depth images* where pixel values represent distances (depths) rather than colors, as is the case with conventional color (RGB) cameras. Many of these devices additionally provide color data, which is why they are called *RGB-D cameras* - this term emphasizes that both, the per-pixel red, green, and blue color information as well as the per-pixel distances are obtained.

These range-sensing cameras can be implemented using various technologies, including structured light (SL), time-of-flight (ToF), and stereo vision. Despite the technical diversity among these sensors, the outcome is equivalent — a per-pixel distance (and color) information. Consequently, throughout this thesis, the technical implementations will not be distinguished, and these devices will simply be referred to as RGB-D or depth cameras. The data will also be referred to as *RGB-D images*, regardless of the specific technology employed. For further information on stereo vision refer to Hartley and Zissermann [HZ00], on SL refer to Zanuttigh et al. [Zan+18], and on ToF sensors refer to Kolb et al. [Kol+10], Grzegorzec et al. [Grz+13], and Hansard et al. [Han+12].

All of these devices typically suffer from various disturbances, including depth shadows, flying pixels, and noisy values. A systematic analysis of these issues was provided, among others, by Sarbolandi et al. [SLK15]. These disturbances pose challenges in numerous applications that depend on accurate depth information. Consequently, there is a growing demand for robust algorithms and improved denoising techniques applied to RGB-D images.

In order to compute 3D points from observed depth information, it is essential to be familiar with the so-called intrinsic camera parameters and therefore to have a basic understanding of the pinhole camera model.

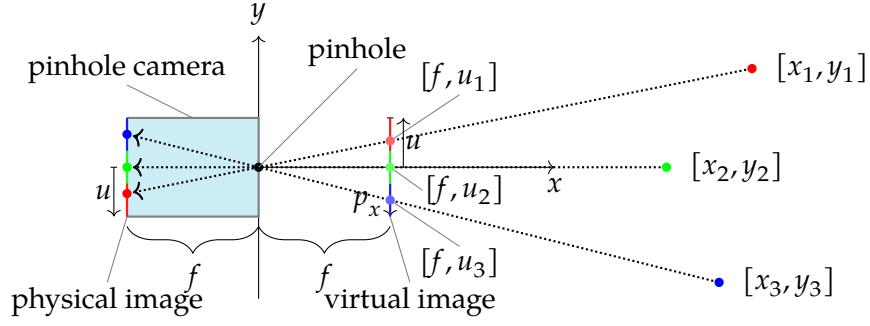


Figure 2.1: Pinhole camera model in 2D. The aperture is a infinitesimally small hole (pinhole). The rays which pass the pinhole are projected onto the physical image, which is a focal length f behind the aperture. The points at camera coordinates $[x_i, y_i]$ are projected to the physical image at $[-f, u_i]$, which corresponds to the virtual image at $[f, u_i]$.

2.1.1 Pinhole Camera Model

Understanding the behavior of light as it travels through a modern camera lens and predicting its location on the camera sensor is, in general, a challenging task. To reduce the complexity of the involved calculations, the assumption of an infinitesimally small aperture (*pinhole*) in place of a lens is made. This simplified representation of a camera is called the *pinhole camera model*.

Pinhole Camera Model in 2D

A reduced 2D version of a pinhole camera is shown in Fig. 2.1. The light from 2D scene points $[x_i, y_i] \in \mathbb{R}^2$ is passing through the pinhole aperture and is projected onto the so-called *physical image* with 1D coordinates $u_i \in \mathbb{R}$. Note that the physical image is a upside-down version of the captured scene. For further simplifications, only the so-called *virtual image* will be considered subsequently, which has the same orientation as the scene (cf. Fig. 2.1). Nevertheless, it is important to emphasize that the coordinates $u_i \in \mathbb{R}$ remain unaffected and are consistent in the physical and virtual images.

The relation between 2D points in camera coordinates $[x, y] \in \mathbb{R}^2$ and their projected coordinates $u \in \mathbb{R}$ is given through the definition of tangent, i.e. $\tan \phi = \frac{\text{opposite}}{\text{adjacent}}$. Since the angle of the light ray through $[f, u]$ and $[x, y]$ is equal, it must be

$$\frac{u}{f} = \frac{y}{x} . \quad (2.1)$$

Thus, given the *focal length* f and the position $[x, y] \in \mathbb{R}^2$ (in camera coordinates) of a point in the scene, it is possible to compute the projected image coordinate u . The final pixel coordinate p_x can be obtained as

$$p_x = \frac{w}{l} u + c_x , \quad (2.2)$$

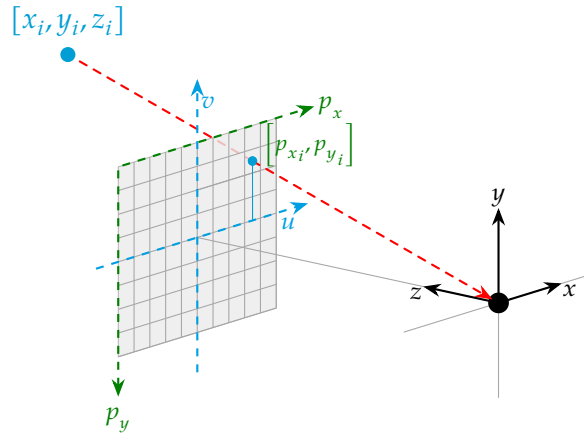


Figure 2.2: Pinhole camera model in 3D. Only the virtual image is sketched.

where w is the width, i.e. the overall number of pixels, l is the (metric) length of the sensor and c_x is the center of the camera in pixels, called *principal point*. The two equations (2.1) and (2.2) can be simplified to

Principal point

$$p_x = f_x \frac{y}{x} + c_x , \quad (2.3)$$

with $f_x = \frac{wf}{l}$ called the *focal length*. Overall, the *intrinsic camera parameters* $f_x, c_x \in \mathbb{R}$ can be used to compute the 1D projection p_x of a 2D point $[x, y] \in \mathbb{R}^2$ in camera coordinates.

Focal length

It is worth to note that equation (2.3) can be used for *back-projection*, i.e. the computation of the 2D position $[x, y] \in \mathbb{R}^2$ of pixel p_x . Therefore the horizontal distance x is needed (e.g. from a depth camera) to compute

Back projection

$$y = \frac{p_x - c_x}{f_x} x . \quad (2.4)$$

Pinhole Camera Model in 3D

The generalization to 3D is straight forward, and shown in Fig. 2.2. In contrast to the 2D model, the sensor has two dimensions and thus the *intrinsic camera parameters* consist of the four values f_x, f_y, c_x , and c_y . Similar to the 2D model, any point $[x, y, z]^T \in \mathbb{R}^3$ in camera coordinates can be projected to pixel coordinates via

Intrinsic camera parameters

$$p_x = f_x \frac{x}{z} + c_x \quad (2.5)$$

$$p_y = f_y \frac{y}{z} + c_y . \quad (2.6)$$

By defining the (*intrinsic*) *camera matrix* $C \in \mathbb{R}^{3 \times 3}$ as

Camera matrix

$$C := \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (2.7)$$

*Camera projection
function $\pi(\cdot)$*

and defining the camera projection function $\pi : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ as

$$\pi([x, y, z]^\top) := \begin{bmatrix} x/z \\ y/z \end{bmatrix}, \quad (2.8)$$

equations (2.5) and (2.6) can be rewritten as

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \pi \left(C \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} \right). \quad (2.9)$$

Pinhole camera model

This equation is the mathematical description of the pinhole camera model in 3D, which is used throughout this thesis for camera related computations.

As in the 2D case (cf. equations (2.3) and (2.4)) it is possible to compute back-projection of equation (2.9) (i.e. of equations (2.5) and (2.6)). Let $\mathcal{D}(p_x, p_y)$ be the horizontal distance $z \in \mathbb{R}_0^+$ at pixel $[p_x, p_y]$ obtained from a depth camera, then equations (2.5) and (2.6) can be used to compute the 3D point $[x, y, z]^\top \in \mathbb{R}^3$ as follows (cf. equation (2.4)):

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{p_x - c_x}{f_x} \\ \frac{p_y - c_y}{f_y} \\ 1 \end{bmatrix} \mathcal{D}(p_x, p_y) \quad (2.10)$$

$$= \begin{bmatrix} \frac{1}{f_x} & 0 & -\frac{c_x}{f_x} \\ 0 & \frac{1}{f_y} & -\frac{c_y}{f_y} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \mathcal{D}(p_x, p_y) \quad (2.11)$$

$$= C^{-1} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \mathcal{D}(p_x, p_y) \quad (2.12)$$

This back-projection of the pinhole camera model will be used to compute point maps from depth images, as discussed in section 2.1.3.

2.1.2 Camera Calibration

In this thesis, the intrinsic camera parameters $f_x, f_y, c_x, c_y \in \mathbb{R}$ were provided by the respective datasets that were used. However, in general, these values are unknown, and a process called *camera calibration* must be performed to obtain these values. This procedure involves capturing a known calibration object (such as a checkerboard) from different perspectives. By establishing correspondences between known 3D points on the object and their 2D projections in the images, this allows for an estimation of the intrinsic camera parameters, as well as the determination of the camera pose relative to the calibration object, referred to as *extrinsic camera parameters*. Further details can be found a.o. in the work of Zhang [Zha00].

Camera calibration

Extrinsic camera parameters

2.1.3 Point and Normal Maps

Let Ω be the set of pixels $[p_x, p_y] \in \Omega$, and $\mathcal{D} : \Omega \rightarrow \mathbb{R}_0^+$ be a depth image from a RGB-D camera with known depth camera matrix $C \in \mathbb{R}^{3 \times 3}$. The back-projection in equation (2.12) yields the *vertex map* or *point map*, i.e. an image with 3D points in camera coordinates per pixel (cf. Fig. 2.2), denoted as $\mathcal{P} : \Omega \rightarrow \mathbb{R}^3$

Point map

$$\mathcal{P}(p_x, p_y) := C^{-1} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \cdot \mathcal{D}(p_x, p_y) . \quad (2.13)$$

From equation (2.9) and the fact that the point map in equation (2.13) represents a 3D surface, the (non-normalized) normal vectors of this surface can be derived as follows

Surface normals

$$\mathbf{n} = \frac{\partial \mathcal{P}}{\partial x} \times \frac{\partial \mathcal{P}}{\partial y} \quad (2.14)$$

$$= \left(\frac{\partial \mathcal{P}}{\partial p_x} \frac{\partial p_x}{\partial x} \right) \times \left(\frac{\partial \mathcal{P}}{\partial p_y} \frac{\partial p_y}{\partial y} \right) \quad (2.15)$$

$$= \left(\frac{\partial \mathcal{P}}{\partial p_x} \times \frac{\partial \mathcal{P}}{\partial p_y} \right) \cdot \underbrace{\left(\frac{\partial p_x}{\partial x} \frac{\partial p_y}{\partial y} \right)}_{\in \mathbb{R} \text{ (scalar)}} . \quad (2.16)$$

These normal vectors are then normalized to yield $\vec{\mathbf{n}} \in \mathbb{S}^2 = \{\mathbf{n} : \mathbb{R}^3 : \|\mathbf{n}\|_2 = 1\}$, thus the scalar on the right hand-side of equation (2.16) can be neglected. Furthermore, since $\mathcal{P}$ is an image, i.e. it is discrete in p_x and p_y , the partial derivatives $\partial/\partial p_x$ and $\partial/\partial p_y$ in equation (2.16), become finite differences. The *normal map* $\mathcal{N} : \Omega \rightarrow \mathbb{S}^2$ can thus be computed e.g. with finite forward differences as:

Normal map

$$\mathbf{n}(p_x, p_y) := (\mathcal{P}(p_x+1, p_y) - \mathcal{P}(p_x, p_y)) \times (\mathcal{P}(p_x, p_y+1) - \mathcal{P}(p_x, p_y)) \quad (2.17)$$

$$\mathcal{N}(p_x, p_y) := \frac{\mathbf{n}(p_x, p_y)}{\|\mathbf{n}(p_x, p_y)\|_2} . \quad (2.18)$$

Equations (2.13) and (2.18) will be used throughout this thesis to compute the point map and the normal map, respectively.

2.2 MATHEMATICAL FUNDAMENTALS

This section gives an overview of fundamental mathematical aspects, which are needed throughout this thesis. Subsequently, only the necessary definitions integral to this thesis are discussed. For a detailed introduction, refer to the works of Kelley [Kel75], Rockafellar [Roc70], Boyd and Vandenberghe [BV04], and/or Bertsekas [Ber09].

2.2.1 Norms

A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is called a norm if it satisfies all of the following conditions (cf. [Roc70, pp. 130–131]):

- f is non-negative: $f(\mathbf{x}) \geq 0$, $\forall \mathbf{x} \in \mathbb{R}^n$
- f definite: $f(\mathbf{x}) = 0$ iff $\mathbf{x} = 0$
- f is homogeneous: $f(k\mathbf{x}) = |k|f(\mathbf{x})$, $\forall \mathbf{x} \in \mathbb{R}^n, \forall k \in \mathbb{R}$
- f satisfies triangle inequality: $f(\mathbf{x} + \mathbf{y}) \leq f(\mathbf{x}) + f(\mathbf{y})$, $\forall \mathbf{x}, \mathbf{y} \in \mathbb{R}^n$

Norm Throughout this thesis (vector) norms are denoted as $\|\mathbf{x}\|_p$ (cf. [BV04, p. 635])

$$\|\mathbf{x}\|_p := \left(\sum_{i=1}^n |x_i|^p \right)^{1/p} . \quad (2.19)$$

Remark 2.1

It can be shown that the above conditions are satisfied only for $p \geq 1$ (cf. [BV04, p. 635]). Hence, for $0 \leq p < 1$, these functions are referred to as *pseudo-norms*. For $p = 0$ it is also called “norm” in literature and throughout this thesis.

The following (vector) norms are used in this thesis:

$$L_1 \text{ norm (Manhattan norm)} \quad \|\mathbf{x}\|_1 := |x_1| + |x_2| + \dots + |x_n| \quad (2.20)$$

$$L_2 \text{ norm (Euclidean norm)} \quad \|\mathbf{x}\|_2 := \sqrt{x_1^2 + x_2^2 + \dots + x_n^2} \quad (2.21)$$

$$L_\infty \text{ norm (Max norm)} \quad \|\mathbf{x}\|_\infty := \max(|x_1|, |x_2|, \dots, |x_n|) , \quad (2.22)$$

as well as the L_0 “norm”, which equals the number of non-zero elements of a vector

$$L_0 \text{ “norm”} \quad \|\mathbf{x}\|_0 := \#\{i \mid x_i \neq 0\} . \quad (2.23)$$

Remark 2.2

The L_1 and L_2 norms are obtained from equation (2.19) by setting $p = 1$ and $p = 2$, respectively. In contrast, L_∞ and L_0 result from the limits $p \rightarrow \infty$ and $p \rightarrow 0$, respectively (cf. [BV04, p. 635]).

2.2.2 Definitions from Set Theory

The ϵ ball of $\|\cdot\|_p$ with radius $\epsilon > 0$ and center $\mathbf{c} \in \mathbb{R}^n$, denoted as $\epsilon\mathbb{B}_p(\mathbf{c})$, is the set of all vectors $\mathbf{x} \in \mathbb{R}^n$ with L_p distance to center $\mathbf{c}$ being lower or equal to ϵ : *ϵ ball*

$$\epsilon\mathbb{B}_p(\mathbf{c}) := \{\mathbf{x} : \|\mathbf{x} - \mathbf{c}\|_p \leq \epsilon\} . \quad (2.24)$$

For $\epsilon = 1$ and center at origin $\mathbf{c} = \mathbf{0}$ the ϵ ball is referred to as *unit ball* ([BV04, p. 634]). The *neighborhood* of a vector $\mathbf{x} \in \mathbb{R}^n$ is defined as the ϵ ball with $p = 2$ throughout this thesis (cf. [Roc70, p. 43] and [Kel75, p. 38]). *Unit ball*
Neighborhood

Let $\mathbb{X} \subset \mathbb{R}^n$ be an arbitrary set. A point $\mathbf{b} \in \mathbb{R}^n$ is called *boundary point* of set $\mathbb{X}$ if and only if each neighborhood of $\mathbf{b}$ intersects both $\mathbb{X}$ as well as $\mathbb{R}^n \setminus \mathbb{X}$ (cf. [Kel75, p. 45]). The set of all boundary points of $\mathbb{X}$ is called *boundary* or *jump set* of $\mathbb{X}$, denoted $\partial\mathbb{X}$. A set $\mathbb{X}$ which contains its boundary, i.e. $\partial\mathbb{X} \subseteq \mathbb{X}$, is called *closed set* and a set $\mathbb{X}$ which is disjoint from its boundary, i.e. $\mathbb{X} \cap \partial\mathbb{X} = \emptyset$, is called *open set* (cf. [Kel75, p. 46]). The *closure* of a set $\mathbb{X}$ is the union of $\mathbb{X}$ and its boundary (cf. [Kel75, p. 46]) *Boundary point*
Boundary
Closed set
Open set
Closure

$$\text{cl}(\mathbb{X}) := \mathbb{X} \cup \partial\mathbb{X} . \quad (2.25)$$

The *interior* of $\mathbb{X}$ is the set without its boundary (cf. [Kel75, p. 46]) *Interior*

$$\text{int}(\mathbb{X}) := \mathbb{X} \setminus \partial\mathbb{X} . \quad (2.26)$$

Remark 2.3

From the above definitions follows (cf. [Ber09, proposition A.2.4]):

A set $\mathbb{X} \subset \mathbb{R}^n$ is closed $\Leftrightarrow \mathbb{X} = \text{cl}(\mathbb{X}) \Leftrightarrow \mathbb{R}^n \setminus \mathbb{X}$ is open.

A set $\mathbb{X} \subset \mathbb{R}^n$ is open $\Leftrightarrow \mathbb{X} = \text{int}(\mathbb{X}) \Leftrightarrow \mathbb{R}^n \setminus \mathbb{X}$ is closed.

Fig. 2.3(a) provides a visual representation of a set's boundary point, boundary, and interior.

The set $\mathbb{X}$ is called *convex* if *Convex set*

$$\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \in \mathbb{X}, \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}, \forall \lambda \in [0, 1] , \quad (2.27)$$

i.e. if all points on a straight line between $\mathbf{x}_1 \in \mathbb{X}$ and $\mathbf{x}_2 \in \mathbb{X}$ are also elements of $\mathbb{X}$ - cf. Fig. 2.3(b). A set which is not convex is called *non-convex*. The linear combination of all points of $\mathbb{X}$ is called *convex hull* of a set, denoted $\text{conv}(\mathbb{X})$ - cf. Fig. 2.3(c): *Non-convex set*
Convex hull

$$\text{conv}(\mathbb{X}) := \{\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2 \mid \forall \mathbf{x}_1, \mathbf{x}_2 \in \mathbb{X}, \forall \lambda \in [0, 1]\} . \quad (2.28)$$

Remark 2.4

The convex hull of $\mathbb{X}$ is the smallest convex superset of $\mathbb{X}$ - cf. [BV04, p. 24].

A set $\mathbb{X}$ is convex $\Leftrightarrow \mathbb{X} = \text{conv}(\mathbb{X})$.

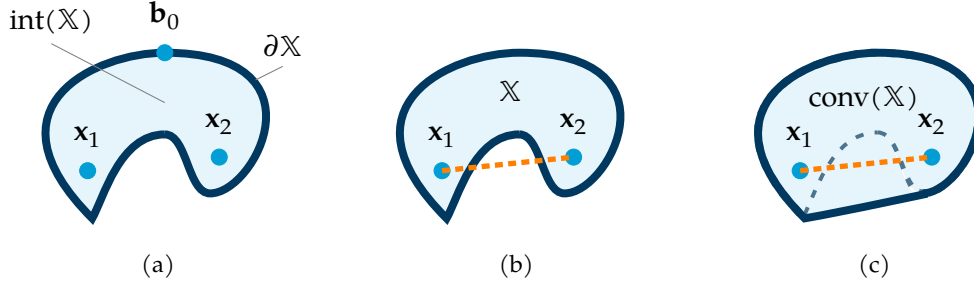


Figure 2.3: A set $X \subset \mathbb{R}^2$, with boundary ∂X , interior $\text{int}(X)$, boundary point $b_0 \in \partial X$, and points $x_1, x_2 \in \text{int}(X)$, is shown in (a). This set is non-convex, since there is (at least) one line between $x_1, x_2 \in X$ which is partially not in X (cf. the orange dashed line), as shown in (b). The convex hull of X is shown in (c).

Convexity is closed under intersection, i.e. given two convex sets K_1 and K_2 , then the intersection $K_1 \cap K_2$ is also a convex set - cf. [Ber09, proposition 1.1.1]. This allows for an alternative definition of the convex hull of a set X , described as the intersection of all convex supersets K_i of X :

$$\text{conv}(X) = \bigcap_{\substack{X \subseteq K_i \\ K_i \text{ is convex}}} K_i . \quad (2.29)$$

It is possible to define a tangential hyperplane at a boundary point $b \in \partial X$ with normal $n \in \mathbb{S}^{n-1} = \{x \in \mathbb{R}^n \mid \|x\|_2 = 1\}$, pointing away from set X (cf. Fig. 2.4(a)):

$$\overline{\mathcal{H}}_i(b) := \{x \mid n_i^\top x = n_i^\top b, n_i \in \mathbb{S}^{n-1}, x \in \mathbb{R}^n\} . \quad (2.30)$$

*Supporting
hyperplane*

Such a tangential hyperplane is called *supporting hyperplane* of the set X , if X is a subset of the halfspace $\mathcal{H}_i(b)$, i.e. if $X \subseteq \mathcal{H}_i(b)$, with

$$\mathcal{H}_i(b) := \{x \mid n_i^\top x \leq n_i^\top b, n_i \in \mathbb{S}^{n-1}, x \in \mathbb{R}^n\} . \quad (2.31)$$

Remark 2.5

A boundary point $b \in \partial X$ may have none, one single, or infinitely many supporting hyperplanes - cf. boundary points b_3 , b_2 and b_1 in Fig. 2.4(a), respectively.

The alternative definition of the convex hull provided in equation (2.29), along with the convex nature of the closed half-spaces $\mathcal{H}_i(\cdot)$, implies that the intersection of the closed half-spaces of all supporting hyperplanes of a set X , is the closure of the convex hull of X - cf. [Ber09, proposition 1.5.4] and Fig. 2.4(b):

$$\text{cl}(\text{conv}(X)) = \bigcap_{\substack{b \in \partial X \\ X \subseteq \mathcal{H}_i(b)}} \mathcal{H}_i(b) . \quad (2.32)$$

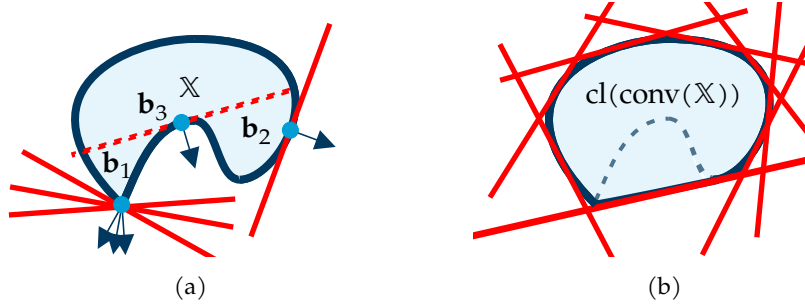


Figure 2.4: A set $\mathbb{X} \subset \mathbb{R}^2$ with supporting hyperplanes at boundary points $\mathbf{b}_1, \mathbf{b}_2 \in \partial\mathbb{X}$ is shown in (a). Note that boundary point $\mathbf{b}_1$ has multiple supporting hyperplanes. Note that $\mathbf{b}_3$ has no supporting hyperplanes since $\mathbb{X}$ is not a subset of the halfspace $\mathcal{H}_i(\mathbf{b}_3)$. The intersection of all closed half-spaces of supporting hyperplanes is a closed convex hull of $\mathbb{X}$, as shown in (b)

Remark 2.6: Supporting Hyperplane Theorem

From equation (2.32), along with remark 2.3 and remark 2.4, follows that for any closed convex set $\mathbb{K}$, there exists a supporting hyperplane for every boundary point $\mathbf{b} \in \partial\mathbb{K}$. This is known as the *supporting hyperplane theorem* - cf. [Ber09, proposition 1.5.1].

Remark 2.7

If a closed set $\mathbb{X}$, with $\text{int}(\mathbb{X}) \neq \emptyset$, has (at least) one supporting hyperplane at every boundary point $\mathbf{b} \in \partial\mathbb{X}$, then the set $\mathbb{X}$ is convex - cf. [BV04, pp. 50—51].

2.2.3 Definitions from Functional Analysis

Let $\mathbb{X} \subseteq \mathbb{R}^n$ be a set and $f : \mathbb{X} \rightarrow [-\infty, +\infty]$ be a function. The *domain* of f is defined as (cf. [Ber09, p. 7]):

$$\text{dom}(f) := \{\mathbf{x} \in \mathbb{X} \mid f(\mathbf{x}) \in]-\infty, +\infty[\} . \quad (2.33)$$

The function f is called *proper* if its domain is non-empty, i.e. $\text{dom}(f) \neq \emptyset$. Otherwise, i.e. $\text{dom}(f) = \emptyset$, the function f is called *improper* (cf. [Ber09, p. 7]).

In the remainder of this section, let $f : \mathbb{X} \rightarrow [-\infty, +\infty]$ denote a proper function. The *graph* of a function f is defined as

$$\text{gra}(f) := \{[\mathbf{x}, f(\mathbf{x})] \in \mathbb{X} \times \mathbb{R} \mid \mathbf{x} \in \text{dom}(f)\} , \quad (2.34)$$

which is a subset of $\mathbb{R}^{n+1}$ - cf. Fig. 2.5(a).

The *epigraph* of a function f is defined as (cf. [Ber09, p. 7]):

$$\text{epi}(f) := \{[\mathbf{x}, k] \in \mathbb{X} \times \mathbb{R} \mid \mathbf{x} \in \text{dom}(f), k \in \mathbb{R}, f(\mathbf{x}) \leq k\} , \quad (2.35)$$

Domain

Proper function
Improper function

Graph

Epigraph

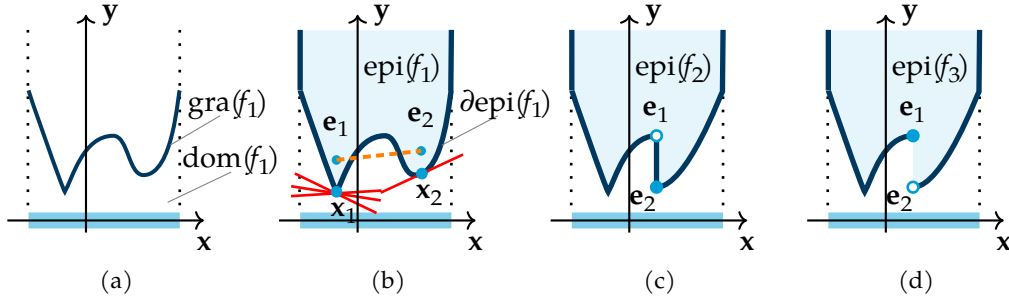


Figure 2.5: This figure shows three different functions $f_i(\mathbf{x}) : \mathbb{X} \rightarrow \mathbb{R}$, with $\mathbb{X} \subset \mathbb{R}$. The graph $\text{gra}(f_1)$ and domain $\text{dom}(f_1)$ are shown in (a). The respective epigraph $\text{epi}(f_1)$ is shown in (b). The function f_1 is not convex, since there is (at least) one line between $\mathbf{e}_1 = [\mathbf{x}_1, y_1] \in \text{epi}(f_1)$ and $\mathbf{e}_2 = [\mathbf{x}_2, y_2] \in \text{epi}(f_1)$, which is not entirely inside the set $\text{epi}(f_1)$. The function f_1 is differentiable at $\mathbf{x}_2$ (since $|\partial f(\mathbf{x}_2)| = 1$) and is not differentiable at $\mathbf{x}_1$. The function f_2 in (c) is l.s.c., since $\text{epi}(f_2)$ is a closed set. If additionally the domain $\text{dom}(f_2)$ is closed, then f_2 is a closed function. The function f_3 in (d) is not l.s.c., since $\text{epi}(f_3)$ is not closed (between $\mathbf{e}_1$ and $\mathbf{e}_2$).

which is the set of all points $[\mathbf{x}, k] \in \mathbb{X} \times \mathbb{R}$ lying on or above the graph of f - cf. Fig. 2.5(b).

Remark 2.8

Note that for any $\mathbf{x} \in \text{dom}(f)$, the point $[\mathbf{x}, f(\mathbf{x})]$ is an element of all three sets: The graph of f , the epigraph of f and the boundary of the epigraph of f - cf. Fig. 2.5(a) and Fig. 2.5(b).

Convex function

The epigraph serves as a tool for analyzing functions and their properties by utilizing concepts from set theory: The function f is called *convex* if its epigraph, $\text{epi}(f)$, is a convex set - cf. Fig. 2.5(b) and [Ber09, definition 1.1.3]. The function f is called *strongly convex* if there exists at least one $\alpha > 0$ such that $f(\mathbf{x}) - \frac{\alpha}{2}\|\mathbf{x}\|_2^2$ is convex.

Strongly convex

l.s.c. function

The function f is called *lower semi-continuous* (l.s.c.) if its epigraph, $\text{epi}(f)$, is a closed set (cf. [Ber09, proposition 1.1.2]). The set of proper, convex and l.s.c. functions $f : \mathbb{X} \rightarrow \mathbb{R}$ on $\mathbb{X}$ is denoted as $\Gamma_o(\mathbb{X})$ throughout this thesis. Fig. 2.5(c) shows a function which is l.s.c., while Fig. 2.5(d) shows a function which is not l.s.c.

$\Gamma_o(\mathbb{X})$

Since $[\mathbf{x}, f(\mathbf{x})] \in \text{epi}(f)$ is element of the boundary of $\text{epi}(f)$ (cf. remark 2.8), it is possible to define a tangential hyperplane through this point. This allows for the definition and analysis of further properties of functions.

Subgradient

Subdifferential

A supporting hyperplane (if it exists) of a function f at $[\mathbf{x}, f(\mathbf{x})] \in \text{epi}(f)$ is called *subgradient* (cf. [Ber09, p. 183]). The set of all subgradients at $f(\mathbf{x})$ is called *subdifferential*, denoted as $\partial f(\mathbf{x})$. The function f is called *subdifferentiable* at $\mathbf{x}$ if the subdifferential $\partial f(\mathbf{x})$ is not empty (cf. [Roc70, p. 215]).

Subdifferentiable $f(\mathbf{x})$

Differentiable $f(\mathbf{x})$

Gradient of $f(\mathbf{x})$

The function f is called *differentiable* at $\mathbf{x}$ if it has only one single subgradient, i.e. $|\partial f(\mathbf{x})| = 1$. This single subgradient is called the *gradient* of f at $\mathbf{x}$, denoted as $\nabla f(\mathbf{x})$ (cf. [Roc70, p. 242]).

The function f is called *differentiable over $\mathbb{X}$* if it is differentiable at every $\mathbf{x} \in \mathbb{X}$. The function is called *differentiable* if it is differentiable for all $\mathbf{x} \in \mathbb{R}^n$ (cf. [Ber09, p. 238]).

Differentiable over $\mathbb{X}$
Differentiable function

If f is differentiable (over an open set $\mathbb{X}$) and $\nabla f(\cdot)$ is continuous at all $\mathbf{x} \in \mathbb{X}$, then f is said to be *continuously differentiable* (over $\mathbb{X}$).

Continuously differentiable function

Fig. 2.5(b) illustrates subgradients of a function f_1 at points $\mathbf{x}_1$ and $\mathbf{x}_2$. It is important to note that f_1 is subdifferentiable at both $\mathbf{x}_1$ and $\mathbf{x}_2$. Furthermore, f_1 is differentiable at $\mathbf{x}_2$ but is not differentiable at $\mathbf{x}_1$.

2.2.4 Indicator Function

The *indicator function* or *characteristic function* of a set $\mathbb{X} \subseteq \mathbb{R}^n$ is defined as:

Indicator function

$$\delta_{\mathbb{X}}(\mathbf{x}) := \begin{cases} 0 & \text{if } \mathbf{x} \in \mathbb{X}, \\ +\infty & \text{otherwise.} \end{cases} \quad (2.36)$$

It serves as a tool to examine sets with concepts from functional analysis, offering the opposite capability of what epigraphs facilitate. For instance, a set $\mathbb{X}$ is convex if and only if $\delta_{\mathbb{X}}(\cdot)$ is a convex function, and a set $\mathbb{X}$ is nonempty if and only if $\delta_{\mathbb{X}}(\cdot)$ is proper.

In the realms of optimization, the indicator function is used to simplify constraints.

Remark 2.9

The following minimization problems are equal (cf. [BV04, p. 69]):

- (i) $\min_{\mathbf{x} \in \mathbb{X}} f(\mathbf{x})$
- (ii) $\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x})$ s.t. $\mathbf{x} \in \mathbb{X}$
- (iii) $\min_{\mathbf{x} \in \mathbb{R}^n} f(\mathbf{x}) + \delta_{\mathbb{X}}(\mathbf{x})$

2.2.5 Support Function

Given an arbitrary non-empty set $\mathbb{X} \subset \mathbb{R}^n$ and a boundary point $\mathbf{b} \in \partial\mathbb{X}$, then in term (2.31) of supporting hyperplanes, equality holds for

$$\sup_{\mathbf{x} \in \mathbb{X}} \mathbf{n}^\top \mathbf{x} = \mathbf{n}^\top \mathbf{b} . \quad (2.37)$$

For non-normalized normal vectors $\mathbf{y} = k\mathbf{n} \in \mathbb{R}^n$ ($k \in \mathbb{R}$), equation (2.37) is still valid and the left hand side is called *support function* $\sigma_{\mathbb{X}} : \mathbb{R}^n \rightarrow \mathbb{R}$ of $\mathbb{X}$:

Support function

$$\sigma_{\mathbb{X}}(\mathbf{y}) := \sup_{\mathbf{x} \in \mathbb{X}} \mathbf{y}^\top \mathbf{x} . \quad (2.38)$$

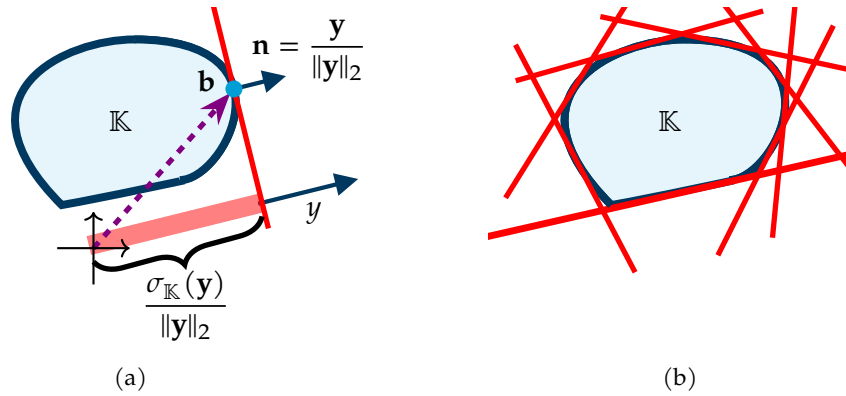


Figure 2.6: This figure illustrates the relation between a closed convex set $\mathbb{K}$ and its support function $\sigma_{\mathbb{K}}(\cdot)$. The geometric interpretation of the support function is shown in (a) - cf. remark 2.10. A closed convex set $\mathbb{K}$ can be characterized either over its elements $\mathbf{x} \in \mathbb{K}$ or over its support function (i.e. its supporting hyperplanes) as shown in (b).

Remark 2.10

The unique feasible solution to equation (2.37) is $\mathbf{x} = \mathbf{b}$, thus $\sigma_{\mathbb{K}}(\mathbf{y}) = \mathbf{y}^\top \mathbf{b} \Leftrightarrow \sigma_{\mathbb{K}}(\mathbf{y}) = \cos(\alpha) \|\mathbf{y}\|_2 \|\mathbf{b}\|_2$ is the (scaled) distance from the origin to a supporting hyperplane with (non-normalized) normal vector $\mathbf{y} \in \mathbb{R}^n$ at boundary point $\mathbf{b} \in \partial \mathbb{K}$ - cf. [Ber09, pp. 88–89] and Fig. 2.6(a).

Dual Representation

The support function $\sigma_{\mathbb{K}}(\cdot)$ allows for a dual representation of closed convex sets. As shown in equation (2.32), any closed convex set $\mathbb{K}$ can be fully characterized by the intersection of all its supporting half-spaces. This offers an alternative approach to representing sets through their boundaries (via the support function) rather than their contents (via the indicator function) - cf. Fig. 2.6(b).

Dual Norm

Dual norm

The support function $\sigma_{\mathbb{B}_p}(\cdot)$ on the unit ball $\mathbb{B}_p := \{\mathbf{x} : \|\mathbf{x}\|_p \leq 1\}$ is called *dual norm* of $\|\cdot\|_p$, denoted as $(\|\cdot\|_p)_*$. The following dual norms can be derived

$$(\|\cdot\|_1)_* = \|\cdot\|_\infty \quad (2.39)$$

$$(\|\cdot\|_2)_* = \|\cdot\|_2 \quad (2.40)$$

$$(\|\cdot\|_\infty)_* = \|\cdot\|_1 \quad (2.41)$$

It can be shown that for $p \geq 1$ it is $(\|\cdot\|_p)_* = \|\cdot\|_q$ with $1/p + 1/q = 1$ - cf. [BV04, pp. 636–637].

2.3 FINITE WEIGHTED GRAPHS

In this work, the primary data structure used for the representation and computation of the solution of image partitioning problem, is the finite weighted graph.

A finite weighted graph is defined as the tuple $G = (V, E, w)$, where V is called set of vertices, $E \subseteq V \times V$ is called set of edges, and w is a weight function $w : E \rightarrow \mathbb{R}^+$.

Graph G

Vertices $v \in V$ initially correspond to pixels $[p_x, p_y] \in \Omega$ of an input image, as shown in Fig. 2.7(a) and Fig. 2.7(b). For the sake of simplicity, in the remainder of this work, images and maps will be defined and accessed via pixels, i.e. $\mathcal{I}(p_x, p_y)$, as well as via the corresponding vertex, i.e. $\mathcal{I}(v)$, with $\mathcal{I}(p_x, p_y) = \mathcal{I}(v)$.

Vertices V

Edges $[v, u] \in E$ define connections between two vertices $v, u \in V$. In this work each vertex v initially is connected to its horizontal neighbors, as well as its vertical neighbors, which is called *4-neighborhood connectivity* - cf. Fig. 2.7(a) and Fig. 2.7(b). The utilization of finite weighted graphs gives the flexibility of removing edges, which will be used to ignore pixels with missing depth values, and for partitioning via connected components (see below).

Edges E

Weights $w : E \rightarrow \mathbb{R}_+$ assign precomputed positive values to edges, and are used for the optimization procedure. For $[v, u] \in E$ weights are defined as

Weights w

$$w(v, u) := \exp(-\|\mathcal{I}(u) - \mathcal{I}(v)\|_2^2), \quad (2.42)$$

which assigns a high value (≈ 1) to edges connecting vertices with similar color values and a low value (≈ 0) to edges connecting vertices with distinct color values.

Remark 2.11

From the definition of weights in equation (2.42) follows the symmetry

$$w(v, u) = w(u, v) \quad (2.43)$$

Let $f : V \rightarrow \mathbb{R}^n$ be an arbitrary function. The weighted graph derivative of f on edge $[v, u] \in E$ is

$$\partial_u^w f(v) := w(v, u) (f(u) - f(v)) . \quad (2.44)$$

Remark 2.12

From the definition of the weighted graph derivative in equation (2.44) follows

$$\partial_u^w f(v) = -\partial_v^w f(u) \quad (2.45)$$

$$\partial_v^w f(v) = \mathbf{0} \quad (2.46)$$

$$f(v) = f(u) \Rightarrow \partial_u^w f(v) = \mathbf{0} \quad (2.47)$$

The weighted gradient operator of the function f (along edge $[v, u] \in E$) is

$$(\nabla_w f)(v, u) := \partial_u^w f(v) . \quad (2.48)$$

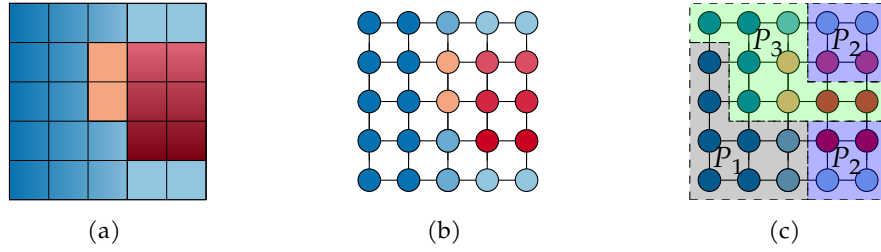


Figure 2.7: A color image is shown in (a), and the corresponding (weighted) finite graph with vertices and edges is shown in (b). An example of a partitioning is given in (c), where partitions P_1 and P_3 in (c) are connected components. Partition P_2 is not a connected component, since there is no path inside P_2 from the top most to the bottom most vertices.

To refer to the gradient of all edges of a vertex $v \in V$ the following notation is used

$$\nabla_w f(v) := (\partial_u^w f(v))_{[v,u] \in E} , \quad (2.49)$$

which is a stacked vector of $\partial_u^w f(v)$ for all edges of a vertex $v \in V$.

The adjoint operator ∇_w^* of the weighted gradient operator is

$$(\nabla_w^* \tilde{f})(v) := \sum_{[v,u] \in E} w(v,u) \tilde{f}(v,u) - \sum_{[u,v] \in E} w(u,v) \tilde{f}(u,v) . \quad (2.50)$$

Remark 2.13

From the definitions of the weighted gradient operator and its adjoint operator follows that both are *linear operators*.

The inner product $\langle f, g \rangle_V$ of two functions over vertices $f, g : V \rightarrow \mathbb{R}^n$ is defined as

$$\langle f, g \rangle_V := \sum_{v \in V} \langle f(v), g(v) \rangle , \quad (2.51)$$

and the inner product $\langle \tilde{f}, \tilde{g} \rangle_E$ of two functions over edges $\tilde{f}, \tilde{g} : E \rightarrow \mathbb{R}^m$ is defined as

$$\langle \tilde{f}, \tilde{g} \rangle_E := \sum_{[v,u] \in E} \langle \tilde{f}(v,u), \tilde{g}(v,u) \rangle . \quad (2.52)$$

Remark 2.14

From equations (2.48) and (2.50) to (2.52) follows

$$\langle \tilde{f}, \nabla_w g \rangle_E = \sum_{[v,u] \in E} \langle \tilde{f}(v,u), w(v,u) (g(u) - g(v)) \rangle \quad (2.53)$$

$$= \sum_{v \in V} \left\langle \sum_{[v,u] \in E} w(v,u) (\tilde{f}(v,u) - \tilde{f}(u,v)), g(v) \right\rangle \quad (2.54)$$

$$= \langle \nabla_w^* \tilde{f}, g \rangle_V \quad (2.55)$$

Let $P_i \subset V$ be partitions (i.e. subsets) of vertices, where for two distinct vertices $v, u \in P_i$ there exists a path (i.e. a set of edges) from v to u (and vice versa), which contains only vertices $\bar{v} \in P_i$. This means that all vertices along this path are also elements of P_i - cf. Fig. 2.7(c). These kind of partitions are called *connected components* or *spatially coherent partitions*. All partitions are pairwise disjoint, i.e. $P_i \cap P_j = \emptyset$ for $i \neq j$, and the union of all partitions is equal to the vertex set, i.e. $\bigcup_{i=1}^N P_i = V$. The set of all partitions is denoted as

Connected components

$$\mathbb{P} = \{P_1, \dots, P_N\} . \quad (2.56)$$

Let $f_{P_i} : V \rightarrow \mathbb{R}^n$ be constant functions over partitions $P_i \in \mathbb{P}$, i.e.

$$f_{P_i}(v) := \begin{cases} \mathbf{c}_i \in \mathbb{R}^n & \text{if } v \in P_i \\ \mathbf{0} = [0, 0, \dots, 0]^\top \in \mathbb{R}^n & \text{otherwise} \end{cases} , \quad (2.57)$$

then the sum over all functions f_{P_i} is a piecewise-constant result over the set of partitions $\mathbb{P}$, denoted as $f_{\mathbb{P}}$:

$$f_{\mathbb{P}}(v) := \sum_{i=1}^N f_{P_i}(v) . \quad (2.58)$$

Remark 2.15

From equation (2.58) follows that $f_{\mathbb{P}}(v) = f_{P_i}(v)$ for $v \in P_i$, and thus $\sum_{i=1}^N \sum_{v \in P_i} \phi(f_{P_i}(v)) = \sum_{v \in V} \phi(f_{\mathbb{P}}(v))$, for arbitrary functionals $\phi : \mathbb{R}^n \rightarrow \mathbb{R}$.

Let $\phi_0 : \mathbb{R}^n \rightarrow \{0, 1\}$ be

$$\phi_0(\mathbf{x}) := \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{0} \\ 1 & \text{otherwise} \end{cases} , \quad (2.59)$$

then the sum of $\phi_0((\nabla_w f_{P_i})(v, u))$ over adjacent neighbors $u \in V$, with $[v, u] \in E$, is the number of *non-zero* elements and thus equal to the L_0 "norm"

$$\|\nabla_w f_{P_i}(v)\|_0 := \sum_{[v, u] \in E} \phi_0((\nabla_w f_{P_i})(v, u)) . \quad (2.60)$$

This definition of the L_0 "norm" of $\nabla_w f_{P_i}(v)$ is equal to the perimeter of partition P_i (cf. [TGB19]):

$$\text{Per}(P_i) = \sum_{[v, u] \in E} \phi_0((\nabla_w f_{P_i})(v, u)) \quad (2.61)$$

$$= \|\nabla_w f_{P_i}(v)\|_0 . \quad (2.62)$$

2.4 VARIATIONAL OPTIMIZATION

In this thesis, the attention is directed towards minimization problems on finite weighted graphs $G = (V, E, w)$, characterized by the following general term:

$$\min_{f(\cdot)} \underbrace{\sum_{v \in V} \phi_D(f(v)) + \lambda \phi_R(\nabla_w f(v))}_{E(\cdot)} , \quad (2.63)$$

Energy

where ϕ_D is called data term, ϕ_R is called regularization term, $f(\cdot)$ is the estimated function, and λ controls the trade-off between the data term and the regularization term. The overall term $E(\cdot)$ is called *energy* or *target function*.

Before exploring specific examples and delving into the solution of equation (2.63), it is essential to revisit two fundamental concepts of optimization theory, namely the convex conjugate and the proximal mapping. For a more detailed discussion, refer to the works of Rockafellar [Roc70], Boyd and Vandenberghe [BV04], Bertsekas [Ber09], Parikh and Boyd [PB14], as well as Chambolle and Pock [CP16]

2.4.1 Convex Conjugate

Let $f : \mathbb{X} \rightarrow \mathbb{R}$ be an arbitrary function, and $\mathbf{n} = [\mathbf{y}^\top, -1]^\top \in \mathbb{R}^{n+1}$ be a (non-normalized) normal vector on $\text{epi}(f)$, which points away from the epigraph. Then the support function (2.38) applied on $\text{epi}(f)$, becomes:

$$\sigma_{\text{epi}(f)}([\mathbf{y}^\top, -1]^\top) = \sup_{[\mathbf{x}, f(\mathbf{x})] \in \text{epi}(f)} [\mathbf{y}^\top, -1] \cdot [\mathbf{x}, f(\mathbf{x})] \quad (2.64)$$

$$\Leftrightarrow \sigma_{\text{epi}(f)}([\mathbf{y}^\top, -1]^\top) = \underbrace{\sup_{\mathbf{x} \in \mathbb{X}} \mathbf{y}^\top \mathbf{x} - f(\mathbf{x})}_{f^*(\mathbf{y})} . \quad (2.65)$$

Convex conjugate

The right-hand side of equation (2.65) is called *convex conjugate* of f , denoted as f^*

$$f^*(\mathbf{y}) := \sup_{\mathbf{x} \in \mathbb{X}} \mathbf{y}^\top \mathbf{x} - f(\mathbf{x}) . \quad (2.66)$$

Like a support function characterises a set, the convex conjugate f^* offers an alternative viewpoint on a function f , based on supporting hyperplanes along the boundary of the epigraph of f . This is a dual perspective on f , which is defined over its (epi)graph - cf. Fig. 2.8(b).

Remark 2.16

Let $f \in \Gamma_o(\mathbb{X})$, $\mathbf{y} \in \mathbb{R}^n$ and $\bar{\mathbf{x}} = \arg \sup_{\mathbf{x} \in \mathbb{X}} \mathbf{y}^\top \mathbf{x} - f(\mathbf{x})$, then $f^*(\mathbf{y}) = \mathbf{y}^\top \bar{\mathbf{x}} - f(\bar{\mathbf{x}})$. The convex conjugate f^* is the vertical distance between the linear function $\mathbf{y}^\top \mathbf{x}$ and $f(\mathbf{x})$ in $\mathbf{x} = \bar{\mathbf{x}}$ - cf. Fig. 2.8(a) and [BV04, p. 91].

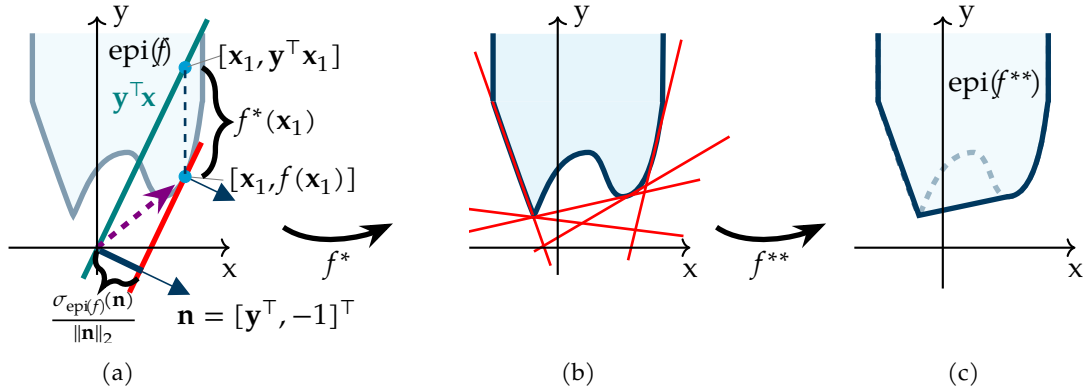


Figure 2.8: This figure shows the relation between a function f , its convex conjugate f^* and its biconjugate f^{**} . The visual interpretations of the support function $\sigma_{\text{epi}(f)}$ on the epigraph of f as well as the convex conjugate f^* are shown in (a). The alternative (dual) representation of the function f over its supporting hyperplanes is shown in (b). The biconjugate $(f^*)^* = f^{**}$ is shown in (c). The epigraph of f^{**} is the closure of the convex hull of the epigraph of f , i.e. $\text{epi}(f^{**}) = \text{cl}(\text{conv}(\text{epi}(f)))$.

The convex conjugate of f^* is called *double conjugate* or *biconjugate*, denoted as f^{**} (cf. [Ber09, pp. 83–84]):

$$f^{**}(\mathbf{x}) := \sup_{\mathbf{y} \in \mathbb{R}^n} \mathbf{x}^\top \mathbf{y} - f^*(\mathbf{y}) , \quad (2.67)$$

where $\mathbf{x}$ is called *primal variable* and $\mathbf{y}$ is called *dual variable*.

Both the convex conjugate f^* and the biconjugate f^{**} have useful properties offering advantages over f itself, making them essential in modern optimization approaches. The relation between f , the convex conjugate f^* and the biconjugate f^{**} is sketched in Fig. 2.8.

Properties

The convex conjugate f^* as well as the biconjugate f^{**} are convex and l.s.c., regardless of the properties of f - cf. [Ber09, pp. 83–84].

From equation (2.66) follows

$$f^*(\mathbf{y}) \geq \mathbf{y}^\top \mathbf{x} - f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{X}, \forall \mathbf{y} \in \mathbb{R}^n \quad (2.68)$$

$$\Leftrightarrow f(\mathbf{x}) + f^*(\mathbf{y}) \geq \mathbf{y}^\top \mathbf{x}, \forall \mathbf{x} \in \mathbb{X}, \forall \mathbf{y} \in \mathbb{R}^n . \quad (2.69)$$

Term (2.69) is known as *Fenchel inequality* (cf. [Ber09, p. 187]).

From equation (2.66) follows also that $f(\mathbf{x}) \geq f^{**}(\mathbf{x})$:

$$f(\mathbf{x}) \geq \mathbf{y}^\top \mathbf{x} - f^*(\mathbf{y}), \forall \mathbf{x} \in \mathbb{X}, \forall \mathbf{y} \in \mathbb{R}^n \quad (2.70)$$

$$\Leftrightarrow f(\mathbf{x}) \geq \underbrace{\sup_{\mathbf{y} \in \mathbb{R}^n} \mathbf{y}^\top \mathbf{x} - f^*(\mathbf{y})}_{f^{**}(\mathbf{x})}, \forall \mathbf{x} \in \mathbb{X} \quad (2.71)$$

Biconjugate

Primal variable
Dual variable

Fenchel inequality

*f^{**} is closed convex
hull of f*

The epigraph of f^{**} is the closed convex hull of the epigraph of f - cf. Fig. 2.8(c). Thus, if $f \in \Gamma_o(\mathbb{X})$, then:

$$f^{**}(\mathbf{x}) = f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{X} , \quad (2.72)$$

and furthermore

$$\mathbf{y} \in \partial f(\mathbf{x}) \Leftrightarrow \mathbf{x} \in \partial f^*(\mathbf{y}) \quad (2.73)$$

- cf. [Ber09, proposition 1.6.1 and pp. 187–188].

Calculus

Let $a \in \mathbb{R} \setminus \{0\}$, then the following relations can be derived from the definition of the convex conjugate function (cf. [BV04, p. 95]).

$$f(\mathbf{x}_1, \mathbf{x}_2) = g(\mathbf{x}_1) + h(\mathbf{x}_2) \Leftrightarrow f^*(\mathbf{y}_1, \mathbf{y}_2) = g^*(\mathbf{y}_1) + h^*(\mathbf{y}_2) \quad (2.74)$$

$$f(\mathbf{x}) = ag(\mathbf{x}) \Leftrightarrow f^*(\mathbf{y}) = ag^*(\mathbf{y}/a) \quad (2.75)$$

$$f(\mathbf{x}) = g(\mathbf{x}) + \mathbf{a}^\top \mathbf{x} + \mathbf{b} \Leftrightarrow f^*(\mathbf{y}) = g^*(\mathbf{y} - \mathbf{a}) - \mathbf{b} \quad (2.76)$$

Examples

From remark 2.9 and the definition of the convex conjugate in equation (2.66), the following relation can be derived:

$$\delta_{\mathbb{X}}(\mathbf{x})^* = \sup_{\mathbf{y} \in \mathbb{R}^n} \mathbf{y}^\top \mathbf{x} - \delta_{\mathbb{X}}(\mathbf{y}) \quad (2.77)$$

$$= \sup_{\mathbf{y} \in \mathbb{X}} \mathbf{y}^\top \mathbf{x} \quad (2.78)$$

$$= \sigma_{\mathbb{X}}(\mathbf{x}) \quad (2.79)$$

i.e. the convex conjugate of the indicator function is the support function and vice-versa.

Likewise from the definition of *dual norms* in section 2.2.5 follows that the convex conjugate of a L_p norm (cf. section 2.2.1) is (cf. [BV04, p. 93]):

$$\|\mathbf{x}\|_p^* = \begin{cases} 0 & \|\mathbf{x}\|_q \leq 1 \\ \infty & \text{otherwise,} \end{cases} \quad (2.80)$$

2.4.2 Proximal Mapping

Moreau envelope

Given a set $\mathbb{X} \subseteq \mathbb{R}^n$ and a function $f : \mathbb{X} \subseteq \mathbb{R}^n \rightarrow \mathbb{R}$, the *Moreau envelope* or *Moreau-Yosida regularization* of this function is defined as (cf. [PB14, p. 135]):

$$M_{\tau f}(\mathbf{x}) := \min_{\mathbf{y} \in \mathbb{X}} \tau f(\mathbf{y}) + \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 . \quad (2.81)$$

The minimizer of the Moreau envelope $M_{\tau f}$ is called *proximal mapping* or *proximity operator*, denoted as $\text{prox}_{\tau f}$ (cf. [PB14, p. 136]): *Proximal mapping*

$$\text{prox}_{\tau f}(\mathbf{x}) := \arg \min_{\mathbf{y} \in \mathbb{X}} \tau f(\mathbf{y}) + \frac{1}{2} \|\mathbf{x} - \mathbf{y}\|_2^2 . \quad (2.82)$$

The Moreau envelope $M_{\tau f}$ is a smoothed and continuously differentiable variant of the function f . The degree of smoothness is determined via the parameter $\tau \in \mathbb{R}$ - cf. Fig. 2.9(b).

Remark 2.17

If f is proper, then (cf. [PB14, pp. 135–136] and Fig. 2.9):

- $M_{\tau f}$ is continuously differentiable, even if f is not
- $\text{dom}(M_{\tau f}) = \mathbb{R}^n$, even if f does not
- $M_{\tau f}(\mathbf{x}) \leq f(\mathbf{x})$, $\forall \mathbf{x} \in \mathbb{X}$
- $\min_{\mathbf{x} \in \mathbb{X}} f(\mathbf{x}) = \min_{\mathbf{x} \in \mathbb{X}} M_{\tau f}(\mathbf{x})$
- $\arg \min_{\mathbf{x} \in \mathbb{X}} f(\mathbf{x}) = \arg \min_{\mathbf{x} \in \mathbb{X}} M_{\tau f}(\mathbf{x})$

Moreau envelopes and proximal mappings are utilized in modern optimization approaches to enable gradient descent steps even when dealing with functions that are not differentiable (cf. remark 2.17). The gradient of $M_{\tau f}$ is (cf. [PB14, p. 136]):

$$\nabla M_{\tau f}(\mathbf{x}) = \frac{1}{\tau} \mathbf{x} - \frac{1}{\tau} \text{prox}_{\tau f}(\mathbf{x}) \quad (2.83)$$

From equation (2.83) follows that the proximal mapping can be interpreted as a gradient descent step, with step size $\tau > 0$, in the Moreau envelope of the function f - cf. Fig. 2.9(a) and (cf. [PB14, pp. 136–137]):

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \tau \nabla M_{\tau f}(\mathbf{x}_k) \quad (2.84)$$

$$\Leftrightarrow \mathbf{x}_{k+1} = \text{prox}_{\tau f}(\mathbf{x}_k) . \quad (2.85)$$

Properties

If $\bar{\mathbf{x}}$ is the minimizer of f , then $\bar{\mathbf{x}}$ is also the minimizer of the Moreau envelope $M_{\tau f}(\mathbf{x})$ (cf. remark 2.17), and thus:

$$\bar{\mathbf{x}} = \text{prox}_{\tau f}(\bar{\mathbf{x}}) , \quad (2.86)$$

which is called *fixed point property* of the proximal mapping (cf. [PB14, p. 126]). *Fixed point*

Another property of the proximal mapping is the *Moreau's identity*, which states that for *Moreau's identity*

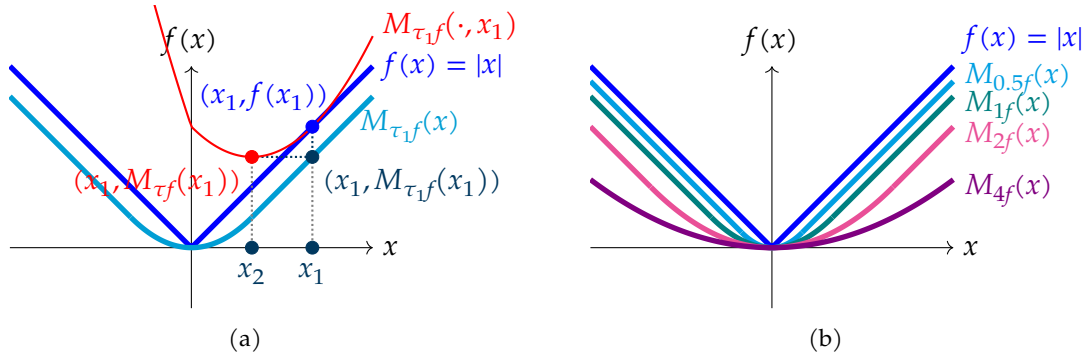


Figure 2.9: This figure shows the function $f(x) = |x|$ and its Moreau envelope. The detailed computation of the Moreau envelope is shown for $x = x_1$ in (a). Note that $x_2 = \text{prox}_{\tau f}(x_1)$. A family of Moreau functions for different τ values is shown in (b).

$f \in \Gamma_o(\mathbb{X})$ the following relation holds (cf. [PB14, p. 133]):

$$\mathbf{x} = \text{prox}_{\tau f}(\mathbf{x}) + \tau \text{prox}_{\frac{1}{\tau} f^*}\left(\frac{1}{\tau} \mathbf{x}\right) \quad (2.87)$$

$$\Leftrightarrow \text{prox}_{\frac{1}{\tau} f^*}\left(\frac{1}{\tau} \mathbf{x}\right) = \mathbf{x} - \text{prox}_{\tau f}(\mathbf{x}) \quad (2.88)$$

This establishes the connection between the proximal mappings of f and its convex conjugate f^* . Furthermore it allows the calculation of the proximity operator of f through the proximity operator of f^* and vice versa.

Calculus

The following relations can be derived from the definition of the proximal mapping (cf. [PB14, pp. 129–132]).

$$f(\mathbf{x}_1, \mathbf{x}_2) = g(\mathbf{x}_1) + h(\mathbf{x}_2) \Leftrightarrow \text{prox}_f(\mathbf{y}_1, \mathbf{y}_2) = [\text{prox}_g(\mathbf{y}_1), \text{prox}_h(\mathbf{y}_2)] \quad (2.89)$$

$$f(\mathbf{x}) = g(\mathbf{x} - \mathbf{b}) \Leftrightarrow \text{prox}_f(\mathbf{y}) = \text{prox}_g(\mathbf{y} - \mathbf{b}) + \mathbf{b} \quad (2.90)$$

$$f(\mathbf{x}) = g(\tau \mathbf{x}) \Leftrightarrow \text{prox}_f(\mathbf{y}) = \frac{1}{\tau} \text{prox}_{\tau^2 g}(\tau \mathbf{y}) \quad (2.91)$$

$$f(\mathbf{x}) = \tau g(\mathbf{x}) \Leftrightarrow \text{prox}_f(\mathbf{y}) = \text{prox}_{\tau g}(\mathbf{y}) \quad (2.92)$$

$$f(\mathbf{x}) = g(\mathbf{x}) + \mathbf{a}^\top \mathbf{x} \Leftrightarrow \text{prox}_f(\mathbf{y}) = \text{prox}_g(\mathbf{y} - \mathbf{a}) \quad (2.93)$$

$$f(\mathbf{x}) = 0 \Leftrightarrow \text{prox}_f(\mathbf{y}) = \mathbf{y} \quad (2.94)$$

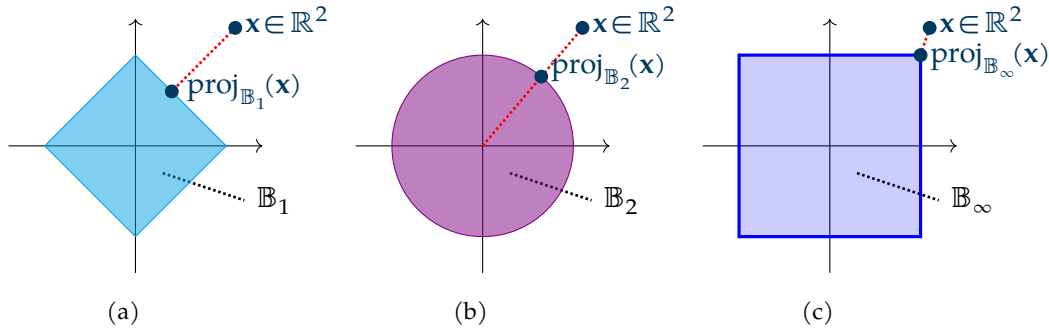


Figure 2.10: Projections of a point $\mathbf{x} \in \mathbb{R}^2$ to the unit balls $\mathbb{B}_1$, $\mathbb{B}_2$ and $\mathbb{B}_\infty$ - cf. equation (2.98) and remark 2.18

Example

Let $f(\mathbf{x}) = \delta_{\lambda\mathbb{B}_p}(\mathbf{x})$ be the *indicator function* (cf. equation (2.36)) of the $\lambda\mathbb{B}_p$ ball (cf. equation (2.24)), then the proximal mapping of this function is

$$\text{prox}_{\tau\delta_{\mathbb{B}_p}}(\mathbf{x}) = \arg \min_{\mathbf{y} \in \mathbb{X}} \tau\delta_{\lambda\mathbb{B}_p}(\mathbf{y}) + \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 \quad (2.95)$$

$$= \arg \min_{\mathbf{y} \in \lambda\mathbb{B}_p} \frac{1}{2}\|\mathbf{x} - \mathbf{y}\|_2^2 \quad (2.96)$$

$$= \begin{cases} \mathbf{x} & \text{if } \mathbf{x} \in \lambda\mathbb{B}_p \\ \text{proj}_{\partial\lambda\mathbb{B}_p}(\mathbf{x}) & \text{if } \mathbf{x} \notin \lambda\mathbb{B}_p \end{cases} \quad (2.97)$$

$$= \text{proj}_{\lambda\mathbb{B}_p}(\mathbf{x}) . \quad (2.98)$$

The proximal mapping of an indicator function of the $\lambda\mathbb{B}_p$ ball, is a projection on this set. The projection function assigns a value on the closest border of the set $\lambda\mathbb{B}_p$ if $\mathbf{x} \notin \lambda\mathbb{B}_p$ and leaves $\mathbf{x}$ unchanged if $\mathbf{x} \in \lambda\mathbb{B}_p$. An illustration of projections on various $\lambda\mathbb{B}_p$ balls is given in Fig. 2.10.

Remark 2.18

It can be shown that the proximal mapping of a indicator function of any closed convex set $\mathbb{K}$ is a projection on this set - cf. [PB14, p. 126]. The projections used in this thesis are (cf. [PB14, pp. 187-188]):

$$\begin{aligned} \left[\text{proj}_{\partial\lambda\mathbb{B}_1}(\mathbf{x}) \right]_i &:= \text{sgn}(\mathbf{x}_i) \cdot \max(|\mathbf{x}_i| - \theta, 0) & (\text{Soft-Thresholding}) & (2.99) \\ \text{s.t. } \sum_{j=1}^n \max(|\mathbf{x}_j| - \theta, 0) &= \lambda \end{aligned}$$

$$\text{proj}_{\partial\lambda\mathbb{B}_2}(\mathbf{x}) := \frac{\lambda\mathbf{x}}{\|\mathbf{x}\|_2} \quad (\text{Normalization}) \quad (2.100)$$

$$\left[\text{proj}_{\partial\lambda\mathbb{B}_\infty}(\mathbf{x}) \right]_i := \min(\lambda, \max(-\lambda, \mathbf{x}_i)) \quad (\text{Clipping}) \quad (2.101)$$

2.4.3 Primal Dual Hybrid Gradient (PDHG)

The *Primal Dual Hybrid Gradient* (PDHG) method, proposed by Chambolle and Pock [CP11], is an iterative algorithm for solving problems of the following shape

$$\min_{\mathbf{f}} G(\mathbf{f}) + F(K\mathbf{f}), \quad (2.102)$$

where $\mathbf{f} \in \mathbb{R}^d$ is the unknown, $K \in \mathbb{R}^{m \times d}$ is a linear operator, and $G : \mathbb{R}^d \rightarrow \mathbb{R}$ as well as $F : \mathbb{R}^m \rightarrow \mathbb{R}$ are two functions which determine the overall optimization problem.

Instead of using $F(\cdot)$ directly, this approach utilizes the biconjugate $F^{**}(\cdot)$ (cf. equation (2.67)) instead, such that the overall problem becomes a so-called *saddle-point problem*:

Saddle-point problem

$$\min_{\mathbf{f}} G(\mathbf{f}) + F(K\mathbf{f}) \quad (2.103)$$

$$\Leftrightarrow \min_{\mathbf{f}} G(\mathbf{f}) + F^{**}(K\mathbf{f}) \quad (2.104)$$

$$\Leftrightarrow \min_{\mathbf{f}} \sup_{\mathbf{y}} G(\mathbf{f}) + \mathbf{y}^\top K\mathbf{f} - F^*(\mathbf{y}) \quad (2.105)$$

This choice is made because the biconjugate function, $F^{**}(\cdot)$, represents the closed convex hull of $F(\cdot)$, as discussed in section 2.4.1. This means that $F^{**}(\cdot)$ is convex as well as l.s.c. and has the same minimizer as $F(\cdot)$. These properties help to reduce the overall complexity of the optimization problem and improve the convergence towards the global optimum [CP11].

To further enhance the optimization algorithm and make it applicable on non-differentiable functions, PDHG utilizes proximal mappings (cf. section 2.4.2), which are equal to a gradient descent of Moreau envelope of a function, as shown in equations (2.83) and (2.85). All in all, the idea of PDHG is to solve equation (2.105) by iteratively performing a proximal gradient ascent with step size $\sigma \in \mathbb{R}_+$ in the *dual variables* $\mathbf{y}(\cdot)$ and a proximal gradient descent with step size $\tau \in \mathbb{R}_+$ in the *primal variables* $\mathbf{f}(\cdot)$ - cf. equation (2.82) as well as equations (2.84) and (2.85). Additionally PDHG contains a relaxation step, which can be steered by the relaxation parameter $\theta \in [0, 1]$, to prevent overly aggressive updates of the primal variable. This leads to the common definition of the PDHG algorithm, called *Alg1* [CP11], shown in algorithm 2.1.

Algorithm 2.1 Primal Dual Hybrid Gradient (PDHG) Algorithm "Alg1"

- 1: Choose $\tau, \sigma > 0$ (with $\tau\sigma\|K\|_{\text{op}}^2 \leq 1$), and relaxation parameter $\theta \in [0, 1]$
 - 2: **repeat**
 - 3: $\mathbf{y}^{k+1} = \text{prox}_{\sigma F^*}(\mathbf{y}^k + \sigma K\bar{\mathbf{f}}^k)$
 - 4: $\mathbf{f}^{k+1} = \text{prox}_{\tau G}(\mathbf{f}^k - \tau K^\top \mathbf{y}^{k+1})$
 - 5: $\bar{\mathbf{f}}^{k+1} = \mathbf{f}^{k+1} + \theta(\mathbf{f}^{k+1} - \mathbf{f}^k)$
 - 6: **until** convergence or $k > k_{\max}$
-

An alternative variant for *strongly convex* problems (cf. section 2.2.3), called Alg2 [CP11], utilizes adaptive step sizes σ_k and τ_k and is shown in algorithm 2.2.

Algorithm 2.2 Primal Dual Hybrid Gradient (PDHG) Algorithm "Alg2"

- 1: Choose initial $\tau_0, \sigma_0 > 0$ (with $\tau_0 \sigma_0 \|K\|_{\text{op}}^2 \leq 1$)
 - 2: **repeat**
 - 3: $y^{k+1} = \text{prox}_{\sigma_k F^*}(y^k + \sigma_k K \bar{f}^k)$
 - 4: $f^{k+1} = \text{prox}_{\tau_k G}(f^k - \tau_k K^\top y^{k+1})$
 - 5: $\theta_k = 1/\sqrt{1 + 2\gamma\tau_k}$
 - 6: $\tau_{k+1} = \theta_k \tau_k$
 - 7: $\sigma_{k+1} = \sigma_k/\theta_k$
 - 8: $\bar{f}^{k+1} = f^{k+1} + \theta_k (f^{k+1} - f^k)$
 - 9: **until** convergence or $k > k_{\max}$
-

Chambolle and Pock [CP11] proved that PDHG Alg1 and Alg2 converge for $\theta = 1$ and step sizes $\tau, \sigma > 0$ with $\tau \sigma \|K\|_{\text{op}}^2 \leq 1$, where the *operator norm* $\|K\|_{\text{op}}^2$ is defined as

Operator norm

$$\|K\|_{\text{op}}^2 = \sup_{\mathbf{x} \neq 0} \frac{\|K\mathbf{x}\|_2^2}{\|\mathbf{x}\|_2^2}. \quad (2.106)$$

Remark 2.19

Since the proximal mapping is used, differentiability is not a requirement for neither $G(\cdot)$ nor $F(\cdot)$. Additionally, since the biconjugate of $F(\cdot)$ (i.e. the closed convex hull of $F(\cdot)$) is used, $F(\cdot)$ itself does not have to be convex. However, the convergence has been formally proven [CP11] for $G, F \in \Gamma_o$, i.e. for proper, convex and l.s.c. functions, and is thus considered a necessary condition for PDHG in literature.

2.4.4 Diagonally Preconditioned PDHG

In the previous algorithms the challenge is to find appropriate step sizes $\tau, \sigma > 0$, such that $\tau \sigma \|K\|_{\text{op}}^2 \leq 1$, in order to guarantee convergence. This criterion requires to compute the operator norm of the linear operator K and thus can be computationally demanding.

In a subsequent publication, Pock and Chambolle [PC11] propose a diagonal preconditioning method, which utilizes diagonal step size matrices $T = \text{diag}(\tau) \in \mathbb{R}^{d \times d}$ and $S = \text{diag}(\sigma) \in \mathbb{R}^{m \times m}$, instead of scalars $\tau, \sigma \in \mathbb{R}_+$. The vectors $\tau = [\tau_1, \dots, \tau_d]$ and $\sigma = [\sigma_1, \dots, \sigma_m]$ are defined as

$$\tau_j := \frac{1}{\sum_{i=1}^m |K_{i,j}|^{2-\alpha}} \quad (2.107)$$

$$\sigma_i := \frac{1}{\sum_{j=1}^d |K_{i,j}|^\alpha} . \quad (2.108)$$

The condition $\tau\sigma\|K\|_{\text{op}}^2 \leq 1$ then becomes $\|S^{\frac{1}{2}}KT^{\frac{1}{2}}\|_{\text{op}}^2 \leq 1$. It can be shown [PC11] that for any $\alpha \in [0, 2]$ it is

$$\left\| S^{\frac{1}{2}}KT^{\frac{1}{2}} \right\|_{\text{op}}^2 = \sup_{\mathbf{x} \neq 0} \frac{\left\| S^{\frac{1}{2}}KT^{\frac{1}{2}}\mathbf{x} \right\|_2^2}{\|\mathbf{x}\|_2^2} \leq 1 . \quad (2.109)$$

This means that by using the diagonal preconditioning as in equation (2.107) and equation (2.108), the PDHG algorithm is guaranteed to converge for $F, G \in \Gamma_o$.

2.4.5 PDHG on Finite Weighted Graphs

The formula for the types of problems solvable by PDHG, shown in equation (2.102), formulated on *finite weighted graphs*, becomes:

$$G(\mathbf{f}) = \sum_{v \in V} \phi_G(f(v)) \quad (2.110)$$

$$F(K\mathbf{f}) = \sum_{v \in V} \phi_F(K_v f(v)) , \quad (2.111)$$

where the unknown $\mathbf{f} \in \mathbb{R}^{|V|n}$, is equal to values of $f(v) : V \rightarrow \mathbb{R}^n$ stacked vertically on top of each other as follows

$$\mathbf{f} = \begin{bmatrix} f(v_1) \\ \vdots \\ f(v_{|V|}) \end{bmatrix} , \quad (2.112)$$

and the linear operator K is a block diagonal matrix

$$K = \begin{bmatrix} K_{v_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & K_{v_2} & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & K_{v_{|V|}} \end{bmatrix} . \quad (2.113)$$

Remark 2.20

The general variational optimization formula on *finite weighted graphs*, shown in equation (2.63), has $K_v = \nabla_w$. This means that the linear operator K can be written as

$$K = \begin{bmatrix} \nabla_w & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \nabla_w & & \vdots \\ \vdots & & \ddots & \mathbf{0} \\ \mathbf{0} & \cdots & \mathbf{0} & \nabla_w \end{bmatrix}. \quad (2.114)$$

Overall the relation between equation (2.63) and equation (2.102) is

$$\phi_G(f(v)) = \phi_D(f(v)) \quad (2.115)$$

$$\phi_F(K_v f(v)) = \lambda \phi_R(\nabla_w f(v)) . \quad (2.116)$$

The *saddle-point problem* in equation (2.105) on *finite weighted graphs* is:

Saddle-point problem

$$\min_{\mathbf{f}} \sup_{\mathbf{y}} G(\mathbf{f}) + \mathbf{y}^\top K \mathbf{f} - F^*(\mathbf{y}) \quad (2.117)$$

$$\Leftrightarrow \min_{f(\cdot)} \sup_{y(\cdot)} \sum_{v \in V} \phi_G(f(v)) + y(v)^\top K_v f(v) - \phi_F^*(y(v)) . \quad (2.118)$$

PDHG Alg1 (cf. algorithm 2.1) on *finite weighted graphs* becomes

Algorithm 2.3 PDHG Alg1 on Finite Weighted Graphs

- 1: Choose $\tau, \sigma > 0$ (with $\tau\sigma\|K\|_{\text{op}}^2 \leq 1$), and relaxation parameter $\theta \in [0, 1]$
 - 2: **repeat**
 - 3: **for each** $v \in V$ **do**: $y^{k+1}(v) = \text{prox}_{\sigma\phi_F^*}(y^k(v) + \sigma K_v \bar{f}^k(v))$
 - 4: **for each** $v \in V$ **do**: $f^{k+1}(v) = \text{prox}_{\tau\phi_G}(f^k(v) - \tau K_v^\top y^{k+1}(v))$
 - 5: **for each** $v \in V$ **do**: $\bar{f}^{k+1}(v) = f^{k+1}(v) + \theta (f^{k+1}(v) - f^k(v))$
 - 6: **until** convergence or $k > k_{\max}$
-

The variant for *strongly convex* problems (cf. algorithm 2.2), called Alg2 [CP11], is shown in algorithm 2.4.

Algorithm 2.4 PDHG Alg2 on Finite Weighted Graphs

- 1: Choose initial $\tau_0, \sigma_0 > 0$ (with $\tau_0 \sigma_0 \|K\|_{\text{op}}^2 \leq 1$)
 - 2: **repeat**
 - 3: **for each** $v \in V$ **do:** $y^{k+1}(v) = \text{prox}_{\sigma_k \phi_F^*}(y^k(v) + \sigma_k K_w \bar{f}^k(v))$
 - 4: **for each** $v \in V$ **do:** $f^{k+1}(v) = \text{prox}_{\tau \phi_G}(f^k(v) - \tau K_v^\top y^{k+1}(v))$
 - 5: $\theta_k = 1/\sqrt{1 + 2\gamma\tau_k}$
 - 6: $\tau_{k+1} = \theta_k \tau_k$
 - 7: $\sigma_{k+1} = \sigma_k/\theta_k$
 - 8: **for each** $v \in V$ **do:** $\bar{f}^{k+1}(v) = f^{k+1}(v) + \theta_k (f^{k+1}(v) - f^k(v))$
 - 9: **until** convergence or $k > k_{\max}$
-

IMAGE PARTITIONING ON FINITE WEIGHTED GRAPHS

Image partitioning is the process of dividing an image into multiple sets of pixels. This is a fundamental task in image processing and computer vision, serving as the foundation for numerous algorithms [CP16], including image denoising [ROF92] and scene flow estimation [Qui+14].

One of the first partitioning models was introduced by Ernst Ising, while working on the calculation of magnetic dipole moments of atomic spins, which were either +1 or -1. Ising derived a two-state model [Isi24] which was later modified by Renfrey Potts [Pot52] to allow for an arbitrary number $N \in \mathbb{N}$ of spin states.

On a finite weighted graph $G = (V, E, w)$ the Potts model reads (cf. remark 2.15)

$$\min_{\mathbb{P}, f_{\mathbb{P}}} E(\mathbb{P}, f_{\mathbb{P}}) = \min_{\mathbb{P}, f_{\mathbb{P}}} \lambda \underbrace{\sum_{P_i \in \mathbb{P}} \text{Per}(P_i)}_{\Phi_R(\mathbb{P})} + \Phi_D(f_{\mathbb{P}}) \quad (3.1)$$

$$= \min_{\mathbb{P}, f_{\mathbb{P}}} \sum_{P_i \in \mathbb{P}} \left(\lambda \underbrace{\text{Per}(P_i)}_{\phi_R(P_i)} + \sum_{v \in P_i} \phi_D(f_{P_i}(v)) \right) \quad (3.2)$$

such that the partitions $P_i \in \mathbb{P}$ (cf. equation (2.56)) cover the whole image domain, i.e. $\bigcup_{P_i \in \mathbb{P}} P_i = \Omega$, and these partitions are pairwise disjoint, i.e. $P_i \cap P_j = \emptyset$ for $i \neq j$.

The functions f_{P_i} and $f_{\mathbb{P}}$ are defined as in equations (2.57) and (2.58), and the scalar $\lambda \in \mathbb{R}$ steers the ratio between data term and regularization term.

Remark 3.1

The piecewise constant Mumford-Shah (PCMS) model [MS89] is a special case of the Potts model, by choosing

$$\phi_D(f_{P_i}(v)) = \|f_{P_i}(v) - g(v)\|_2^2 \quad (3.3)$$

In this thesis, the image partitioning problem is addressed and analyzed on finite weighted graphs in two distinct ways: Through an approximation of the L_0 “norm” via the truncated quadratic “norm”, discussed in section 3.1, and through an algorithm called *Cut Pursuit*, detailed in section 3.2.

3.1 PARTITIONING VIA TRUNCATED QUADRATICS

This section delves into solving the optimization problem given in equation (3.2), with the *truncated quadratic “norm”* as an approximation of the L_0 “norm”. While the main focus is on the piecewise constant Mumford-Shah (PCMS) model on a finite weighted graph $G = (V, E, w)$ (cf. remark 3.1), it is important to emphasize that this method is also applicable for other definitions of $\phi_D(\cdot)$.

3.1.1 Derivation

Utilizing PCMS data term in equation (3.3) along with the L_0 “norm”, the optimization problem in equation (3.2) becomes:

$$\min_{\mathbb{P}, f_{\mathbb{P}}} \sum_{P_i \in \mathbb{P}} \left(\sum_{v \in P_i} \sum_{[v, u] \in E} \lambda \phi_0 \left((\nabla_w f_{P_i})(v, u) \right) + \|f_{P_i}(v) - g(v)\|_2^2 \right) \quad (3.4)$$

The central idea of the method proposed by Strekalovskiy and Cremers [SC14] is to model the edge set of partitions $P_i \in \mathbb{P}$ implicitly as gradients of the piecewise constant result $f_{\mathbb{P}}(\mathbf{x}) = \sum_{i=1}^N f_{P_i}(\mathbf{x})$ (cf. equations (2.57) and (2.58)). Additionally $\phi_0(\cdot)$ is replaced with the *truncated quadratic* function $\phi_{\alpha TQ}(\cdot)$

$$\phi_{\alpha TQ}(\mathbf{x}) := \begin{cases} \alpha \|\mathbf{x}\|_2^2 & \text{if } \alpha \|\mathbf{x}\|_2^2 < 1 \\ 1 & \text{otherwise} \end{cases} \quad (3.5)$$

$$= \min(\alpha \|\mathbf{x}\|_2^2, 1) . \quad (3.6)$$

The function $\phi_{\alpha TQ}(\cdot)$ can be considered as a relaxation of $\phi_0(\cdot)$ because of the following relation:

$$\lim_{\alpha \rightarrow +\infty} \phi_{\alpha TQ}(\mathbf{x}) = \phi_0(\mathbf{x}) \quad \forall \mathbf{x} \in \mathbb{R}^n . \quad (3.7)$$

Following the idea of replacing $\phi_0(\cdot)$ by $\phi_{\alpha TQ}(\cdot)$ in equation (3.4) as well as the relation in remark 2.15, the overall PCMS model becomes:

$$\min_{\mathbb{P}, f_{\mathbb{P}}} \sum_{P_i \in \mathbb{P}} \sum_{v \in P_i} \left(\sum_{[v, u] \in E} \lambda \phi_{\alpha TQ} \left((\nabla_w f_{P_i})(v, u) \right) \right) + \|f_{P_i}(v) - g(v)\|_2^2 \quad (3.8)$$

$$= \min_{f_{\mathbb{P}}} \sum_{v \in V} \left(\sum_{[v, u] \in E} \lambda \phi_{\alpha TQ} \left((\nabla_w f_{\mathbb{P}})(v, u) \right) \right) + \|f_{\mathbb{P}}(v) - g(v)\|_2^2 . \quad (3.9)$$

It is worth to note that partitions $P_i \in \mathbb{P}$ do not appear in equation (3.9), which is a consequence of the implicit definition of the edge set as gradients of $f_{\mathbb{P}}$. This leads to partitions P_i , their boundaries ∂P_i as well as the partition set $\mathbb{P}$ to be all implicitly defined over the gradient of $f_{\mathbb{P}}$.

The shape of equation (3.9) matches the general energy term of solvable problems with PDHG, shown in equation (2.102), by choosing (cf. remark 2.20):

$$G(\mathbf{f}) = \sum_{v \in V} \underbrace{\|f_{\mathbb{P}}(v) - g(v)\|_2^2}_{\phi_D(\cdot)} \quad (3.10)$$

$$(K\mathbf{f}) = \sum_{v \in V} \underbrace{\sum_{[v,u] \in E} \lambda \phi_{\alpha TQ}((\nabla_w f_{\mathbb{P}})(v, u))}_{\phi_R(\cdot)} \quad (3.11)$$

$$K_v = \nabla_w . \quad (3.12)$$

This optimization problem can be solved with PDHG as shown in section 2.4.5. Therefore $\text{prox}_{\tau \phi_D}(\cdot)$, ϕ_R^* and $\text{prox}_{\sigma \phi_R^*}(\cdot)$ must be derived.

The function ϕ_R^* can be computed by using the separable sum property of convex conjugate, shown in equation (2.74), as follows

$$\phi_R^*(\nabla_w f_{\mathbb{P}}(v)) = \sum_{[v,u] \in E} \lambda \phi_{\alpha TQ}^*((\nabla_w f_{\mathbb{P}})(v, u)) . \quad (3.13)$$

Thus $\text{prox}_{\sigma \phi_R^*}(\cdot)$ can be computed by using the separable sum property of the proximal mapping, shown in equation (2.89), as follows

$$\text{prox}_{\sigma \phi_R^*}([y_1, \dots, y_k]) = \left[\text{prox}_{\sigma \phi_{\alpha TQ}^*}(y_1), \dots, \text{prox}_{\sigma \phi_{\alpha TQ}^*}(y_k) \right] , \quad (3.14)$$

where $y_i = (\nabla_w f_{\mathbb{P}})(v, u)$ for $[v, u] \in E$.

The convex conjugate of $\phi_{\alpha TQ}(\cdot)$ is

$$\phi_{\alpha TQ}^*(\mathbf{y}) = \sup_{\mathbf{x} \in \mathbb{R}^n} \mathbf{y}^\top \mathbf{x} - \min(\alpha \|\mathbf{x}\|_2^2, 1) \quad (3.15)$$

$$= \begin{cases} 0 & \text{for } \mathbf{y} = \mathbf{0} \\ +\infty & \text{otherwise} \end{cases} , \quad (3.16)$$

which can be thought of as an indicator function $\delta_{\mathbb{X}}(\cdot)$ of a set $\mathbb{X} = \{\mathbf{0}\}$ containing the zero vector only. As shown in equation (2.98), the proximal mapping of the indicator function of a convex set $\mathbb{X}$ will simply project the input onto that set, thus

$$\text{prox}_{\sigma \phi_{\alpha TQ}^*}(\mathbf{y}) = \mathbf{0} . \quad (3.17)$$

This is a logical consequence of the fact that the underlying *saddle-point problem* of the PDHG algorithm utilizes the biconjugate $\phi_{\alpha TQ}^{**}(\cdot)$, which is the closed convex hull of the function $\phi_{\alpha TQ}(\cdot)$, and the closed convex hull of the truncated quadratic is simply

$$\phi_{\alpha TQ}^{**}(\mathbf{x}) = \mathbf{0} . \quad (3.18)$$

This means that PDHG applied on this problem would behave as if there is no regularization term at all. Thus the optimal solution would be trivially $f_{\mathbb{P}}(v) = g(v)$, which is clearly not desired.

To bypass this problem, Strekalovskiy and Cremers [SC14] propose to utilize *Moreaus's identity*, introduced in equation (2.88), as follows:

$$\text{prox}_{\sigma \phi_{\alpha TQ}^*}(\mathbf{x}) = \mathbf{x} - \sigma \text{prox}_{\frac{1}{\sigma} \phi_{\alpha TQ}}(\mathbf{x}/\sigma) \quad (3.19)$$

This allows to compute $\text{prox}_{\sigma \phi_{\alpha TQ}^*}(\mathbf{x})$ via $\text{prox}_{\sigma \phi_{\alpha TQ}}(\mathbf{x})$, i.e. the convex conjugate is not needed, which overcomes the previously mentioned problem of ignoring the regularization term and thus avoids the trivial solution. However, as mentioned in section 2.4.2 *Moreau's identity* is valid for convex and l.s.c. functions $g \in \Gamma_o(\mathbb{R}^n)$, which is not the case for the function $\phi_{\alpha TQ}(\cdot)$. Therefore equation (3.19) can be considered as an approximation of $\text{prox}_{\sigma \phi_{\alpha TQ}^*}(\cdot)$.

The explicit formula for the proximal operator of $\phi_{\alpha TQ}(\cdot)$ is

$$\text{prox}_{\sigma \phi_{\alpha TQ}}(\mathbf{y}) = \begin{cases} \frac{1}{1+2\sigma\alpha} \mathbf{y} & \text{if } \|\mathbf{y}\|_2^2 \leq \lambda(1+2\sigma\alpha)/\alpha \\ \mathbf{y} & \text{otherwise} \end{cases} \quad (3.20)$$

which, inserted into equation (3.19), leads to

$$\text{prox}_{\sigma \phi_{\alpha TQ}^*}(\mathbf{y}) = \begin{cases} \frac{2\alpha}{\sigma+2\alpha} \mathbf{y} & \text{if } \|\mathbf{y}\|_2^2 \leq \lambda\sigma(\sigma+2\alpha)/\alpha \\ \mathbf{0} & \text{otherwise} . \end{cases} \quad (3.21)$$

In order to obtain piecewise constant results, the limit case $\alpha \rightarrow +\infty$ must be considered, which, along with equation (3.7), finally leads to

$$\lim_{\alpha \rightarrow +\infty} \text{prox}_{\sigma \phi_{\alpha TQ}^*}(\mathbf{y}) = \begin{cases} \mathbf{y} & \text{if } \|\mathbf{y}\|_2^2 \leq 2\lambda\sigma \\ \mathbf{0} & \text{otherwise} \end{cases} \quad (3.22)$$

$$= \text{prox}_{\sigma \phi_0^*}(\mathbf{y}) . \quad (3.23)$$

The function $\text{prox}_{\tau \phi_D}(\cdot)$ can be computed explicitly:

$$\text{prox}_{\tau \phi_D}(\mathbf{x}) = \frac{\mathbf{x} + 2\tau g(\mathbf{x})}{1 + 2\tau} . \quad (3.24)$$

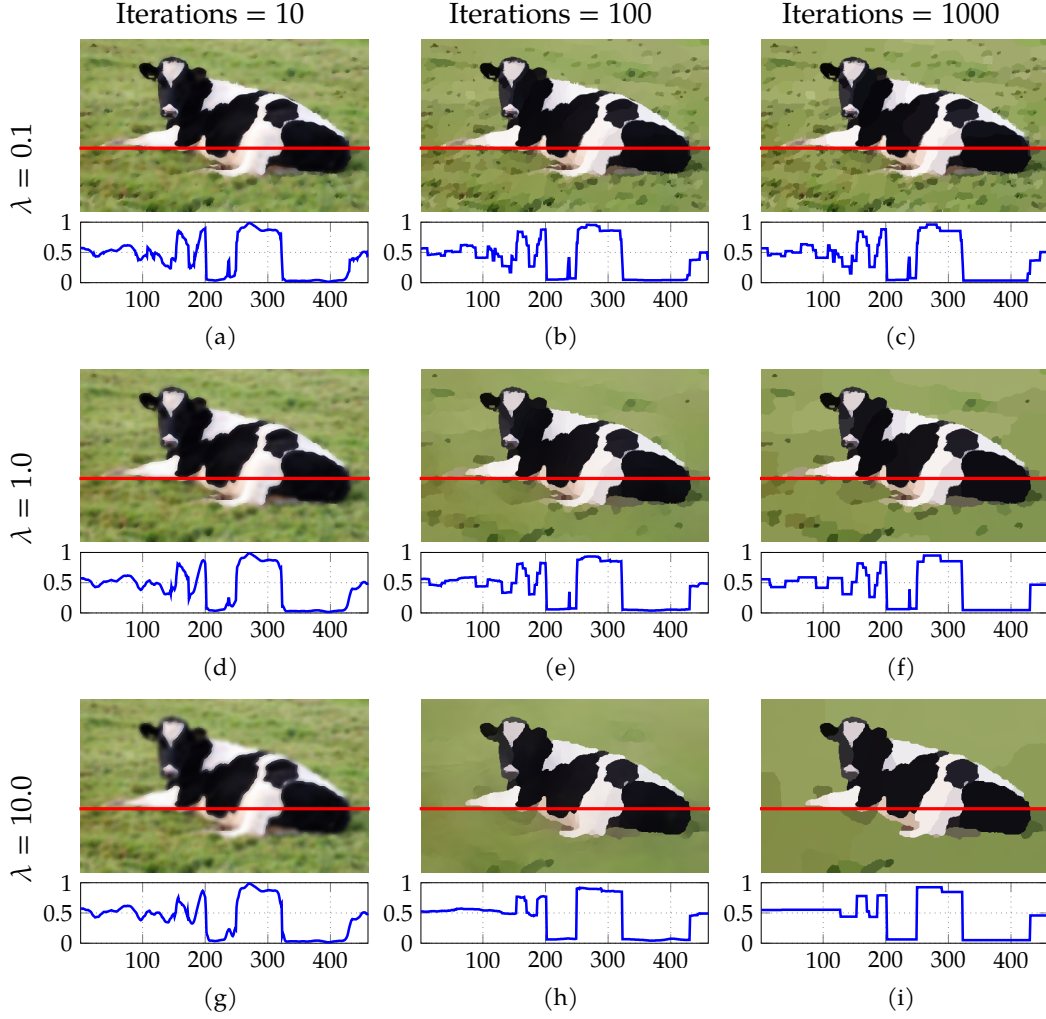


Figure 3.1: Results of partitioning via truncated quadratics for different λ values and iterations.

Given the proximal mappings in equation (3.22) and equation (3.24), the image partitioning problem in equation (3.9) can be solved with PDHG algorithm 2.4. Strekalovskiy and Cremers [SC14] propose $\tau_0 = \frac{1}{4}$ and $\sigma_0 = \frac{1}{2}$, which satisfies the PDHG convergence criterion $\tau_0 \sigma_0 \|K\|_{\text{op}}^2 \leq 1$, since $\|K\|_{\text{op}}^2 < \frac{1}{8}$ [CP11].

3.1.2 Results

The results of the previously derived algorithm for different values of λ and iterations are presented in Fig. 3.1. Notably, a low number of iterations does not yield piecewise constant results, which is a major drawback of this approach. Achieving such results requires a sufficient number of iterations. Furthermore it is worth to note that the parameter λ controls the size of partitions, determining the granularity of the piecewise constant result.

3.2 PARTITIONING VIA CUT PURSUIT

Cut Pursuit is a partitioning algorithm which follows a coarse-to-fine approach. It starts from an initial partitioning $\mathbb{P}_0$, e.g. a trivial partitioning $\mathbb{P}_0 = \{V\}$ where one single partition contains all vertices. The goal of *Cut Pursuit* is to divide a given partitioning $\mathbb{P}_i$ to a partitioning $\mathbb{P}_{i+1}$, such that the overall energy in equation (3.1) is reduced:

$$E(f_{\mathbb{P}_{i+1}}) \leq E(f_{\mathbb{P}_i}) . \quad (3.25)$$

3.2.1 Derivation

Equation (3.25) can be translated to the optimization problem

$$\min_{f_{\mathbb{P}_{i+1}}} E(f_{\mathbb{P}_{i+1}}) - E(f_{\mathbb{P}_i}) \quad (3.26)$$

$$\Leftrightarrow \min_{f_{\mathbb{P}_{i+1}}} \Phi_D(f_{\mathbb{P}_{i+1}}) - \Phi_D(f_{\mathbb{P}_i}) + \Phi_R(\nabla_w f_{\mathbb{P}_{i+1}}) - \Phi_R(\nabla_w f_{\mathbb{P}_i}) . \quad (3.27)$$

The difference of the data terms $\Phi_D(\cdot)$ can be approximated over the first order Taylor polynomial as

$$\Phi_D(f_{\mathbb{P}_{i+1}}) - \Phi_D(f_{\mathbb{P}_i}) \approx \sum_{v \in V} \langle \nabla \phi_D(f_{\mathbb{P}_i}(v)), f_{\mathbb{P}_{i+1}}(v) - f_{\mathbb{P}_i}(v) \rangle , \quad (3.28)$$

and the difference of the regularization terms $\Phi_R(\cdot)$, cf. Figs. 3.2(a), 3.2(b) and 3.2(d), yields

$$\Phi_R(\nabla_w f_{\mathbb{P}_{i+1}}) - \Phi_R(\nabla_w f_{\mathbb{P}_i}) = \lambda \underbrace{\sum_{P_k^{i+1} \in \mathbb{P}_{i+1}} \text{Per}(P_k^{i+1}) - \sum_{P_j^i \in \mathbb{P}_i} \text{Per}(P_j^i)}_{\lambda \cdot \text{length of new partition borders}} . \quad (3.29)$$

The strategy of *Cut Pursuit* is to partition the vertex set V in two partitions B and $\bar{B}$ with $\bar{B} = V \setminus B$. These two sets are then used to divide partitions $P_j^i \in f_{\mathbb{P}_i}$ by intersecting with B and $\bar{B}$, as shown in Figs. 3.2(b) and 3.2(c), as follows

$$\bar{P}_{2j-1}^{i+1} = P_j^i \cap B \quad (3.30)$$

$$\bar{P}_{2j}^{i+1} = P_j^i \cap \bar{B} . \quad (3.31)$$

The new partitions P_k^{i+1} are created from connected components of partitions $\bar{P}_j^{i+1}$ (cf. Fig. 3.2(d)).

Landrieu and Obozinsky [LO17] propose to estimate a real function $f_{\{B, \bar{B}\}} : V \rightarrow [-1, 1]$, such that $f_{\mathbb{P}_{i+1}}(v)$ can be approximated as follows

$$f_{\mathbb{P}_{i+1}}(v) \approx f_{\mathbb{P}_i}(v) + \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix} f_{\{B, \bar{B}\}}(v) , \quad (3.32)$$

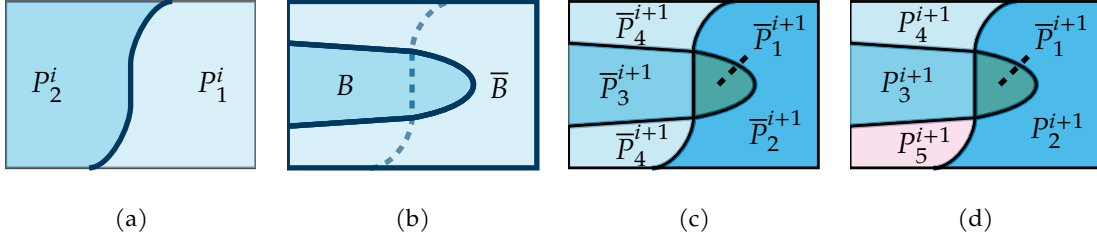


Figure 3.2: These figures sketch the partitioning process of the *Cut Pursuit* [LO17] algorithm. A given partitioning in (a), is refined by a binary partitioning $B, \bar{B}$ in (b). The new partitions in (c) are computed via equations (3.30) and (3.31). The final partitions P_k^{i+1} are computed as connected components of $\bar{P}_j^{i+1}$. The difference of the boundaries of $f_{\mathbb{P}_i} = \{P_1^i, P_2^i\}$ in (a) and $f_{\mathbb{P}_{i+1}} = \{P_1^{i+1}, \dots, P_5^{i+1}\}$ in (d) is the boundary of the binary partitioning in (b).

and therefore the relation in equation (3.28) becomes

$$\Phi_D(f_{\mathbb{P}_{i+1}}) - \Phi_D(f_{\mathbb{P}_i}) \approx \sum_{v \in V} \langle \nabla \phi_D(f_{\mathbb{P}_i}(v)), f_{\mathbb{P}_{i+1}}(v) - f_{\mathbb{P}_i}(v) \rangle \quad (3.33)$$

$$\Leftrightarrow \Phi_D(f_{\mathbb{P}_{i+1}}) - \Phi_D(f_{\mathbb{P}_i}) \approx \sum_{v \in V} \langle \nabla \phi_D(f_{\mathbb{P}_i}(v)), [1, 1, \dots, 1]^\top f_{\{B, \bar{B}\}}(v) \rangle \quad (3.34)$$

$$\Leftrightarrow \Phi_D(f_{\mathbb{P}_{i+1}}) - \Phi_D(f_{\mathbb{P}_i}) \approx \sum_{v \in V} \langle [1, 1, \dots, 1] \cdot \nabla \phi_D(f_{\mathbb{P}_i}(v)), f_{\{B, \bar{B}\}}(v) \rangle. \quad (3.35)$$

Furthermore, since the borders of B and $\bar{B}$ are *new* in the partitioning, the perimeter $\text{Per}(B)$ equals the length of new partition borders (cf. equation (3.29)), such that together with equation (3.32) it is

$$\Phi_R(f_{\mathbb{P}_{i+1}}) - \Phi_R(f_{\mathbb{P}_i}) = \lambda \text{Per}(B) \quad (3.36)$$

$$= \lambda \|\nabla_w f_{\{B, \bar{B}\}}\|_0 \quad (3.37)$$

Inserting equations (3.35) and (3.37) into equation (3.27) leads to the variational binary partitioning problem

$$\min_{f_{\{B, \bar{B}\}}} \sum_{v \in V} \langle [1, 1, \dots, 1] \cdot \nabla \phi_D(f_{\mathbb{P}_i}(v)), f_{\{B, \bar{B}\}}(v) \rangle + \lambda \|\nabla_w f_{\{B, \bar{B}\}}\|_0, \quad (3.38)$$

which can be solved by PDHG [CP16] or by the *Graph Cut* approach of Boykov and Kolmogorov [BK04].

In case of PDHG the L_1 norm is used as a relaxation of the L_0 norm and thus (cf. remark 2.20)

$$G(\mathbf{f}) = \sum_{v \in V} \underbrace{\langle [1, 1, \dots, 1] \cdot \nabla \phi_D(f_{\mathbb{P}_i}(v)), f_{\{B, \bar{B}\}}(v) \rangle}_{\phi_G} \quad (3.39)$$

$$F(K\mathbf{f}) = \sum_{v \in V} \underbrace{\lambda \|\nabla_w f_{\{B, \bar{B}\}}(v)\|_1}_{\phi_F} \quad (3.40)$$

$$K_v = \nabla_w. \quad (3.41)$$

From the *affine addition* property, cf. equation (2.93), of the proximal operator follows

$$\text{prox}_{\tau\phi_G}(\mathbf{x}) = \mathbf{x} - \tau[1, 1, \dots, 1] \cdot \nabla\phi_D(f_{\mathbb{P}_i}(v)) \quad (3.42)$$

and

$$\text{prox}_{\sigma\phi_F^*}(\mathbf{y}) = \text{proj}_{\lambda\mathbb{B}_\infty}(\mathbf{y}) . \quad (3.43)$$

The projection function $\text{proj}_{\lambda\mathbb{B}_\infty}(\mathbf{y})$ is defined as in equation (2.98) and remark 2.18.

The whole *Cut Pursuit* algorithm operates as shown in algorithm 3.1.

Algorithm 3.1 Cut Pursuit Algorithm

- 1: Initial partitioning $\mathbb{P}_0 = \{V\}$
 - 2: **repeat**
 - 3: $\{B, \bar{B}\} \leftarrow$ solve binary partitioning in equation (3.38)
 - 4: $\bar{P}_{2j-1}^{i+1}, \bar{P}_{2j}^{i+1} \leftarrow$ compute new partitions $P_j^i \cap B$ and $P_j^i \cap \bar{B}$
 - 5: $P_k^{i+1} \leftarrow \text{connComp}(\bar{P}_j^{i+1})$
 - 6: $f_{\mathbb{P}_{i+1}} \leftarrow$ solve Φ_D per partition P_k^{i+1}
 - 7: **until** convergence ($E(f_{\mathbb{P}_{i+1}}) \approx E(f_{\mathbb{P}_i})$) or $i > i_{\max}$
-

3.2.2 Results

The results of *Cut Pursuit* algorithm for different values of λ and iterations are presented in Fig. 3.3. It can be observed that similar to the variational algorithm in section 3.1 the value of λ controls the size of partitions, determining the granularity of the piecewise constant result. Furthermore this algorithm provides piecewise constant results after each iteration. However, it is worth noting that a single iteration of *Cut Pursuit* involves solving the *graph cut* problem, making it computationally as expensive as multiple iterations of the truncated quadratic approach in section 3.1.

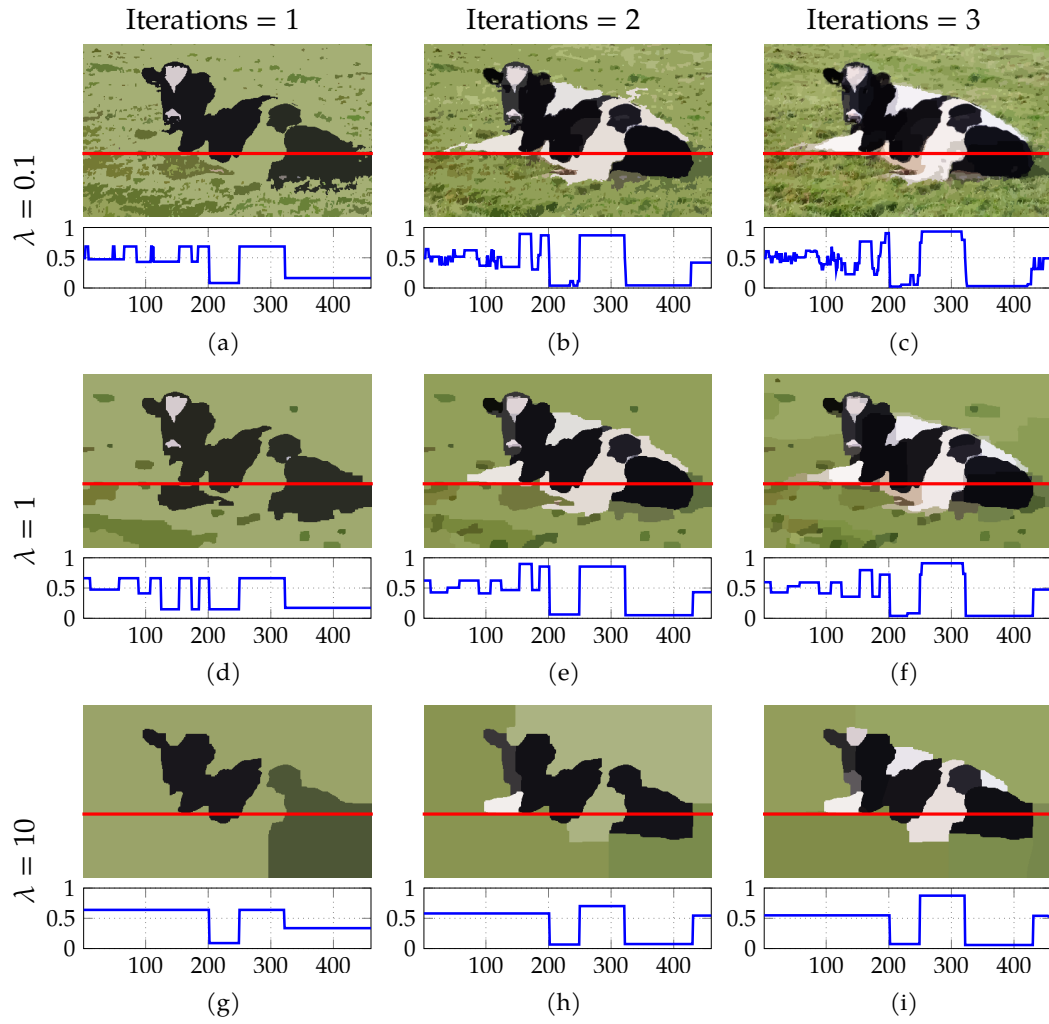


Figure 3.3: Results of partitioning via *Cut Pursuit* for different λ values and iterations.

PIECEWISE RIGID SCENE FLOW ESTIMATION

In this chapter, a variational approach for the estimation of scene flow from RGB-D images is presented. The derived model enforces piecewise rigid motion through decomposition into rotational and translational motion parts, which are regularized by the L_0 “norm”. This facilitates implicit motion segmentation in a joint energy minimization problem. To bypass computational complexities, involved in the L_0 “norm” regularization, the *truncated quadratic norm* is used as a substitute. The overall optimization problem is solved efficiently by the PDHG algorithm, similar to the variational minimal partitioning approach in section 3.1.

This chapter is organized as follows. It begins with an introduction that provides the motivation behind this topic. Following this, a review of related work is presented to contextualize the research within the existing literature. Next, details of the proposed approach are presented, explaining its methodology and framework. Finally, the proposed approach is evaluated, and the results are discussed to highlight its effectiveness.

The approach discussed in this chapter was published [GGK19] and presented at the International Conference on Intelligent Robots and Systems (IROS) 2019 in Macau, China.

4.1 MOTIVATION

Motion estimation, whether it involves the movement of the camera or the motion of (rigid or deformable) objects, holds significant interest across various research fields due to its wide range of applications, particularly in robotics. Tasks such as object tracking, gesture recognition, and diverse forms of robotic interaction with the dynamic real world require a comprehensive understanding of these motions. However, even though the methods to estimate camera motion and the motion of 3D points share common challenges, they are addressed using different approaches. Each of these methods comes with its own unique set of advantages and disadvantages.

Estimation of the camera pose from images is referred to as visual odometry. This is one of the central techniques in robotics for simultaneous localization and mapping (SLAM) [DB06], allowing online 3D scene reconstruction and navigation of unmanned vehicles in unknown terrains. In the case of scene dynamics, however, the assumption that the scene is static leads to faulty pose estimations, which is a prevailing drawback of *state of the art* SLAM approaches [Zol+18].

In contrast to visual odometry, scene flow describes the inter-frame three-dimensional motion for each single 3D point. The scene flow problem is inherently ill-posed, and it is thus necessary to incorporate prior knowledge via regularization [YX17]. Smoothness regularization is commonly applied, yet as all points are handled equally, static and dynamic scene-parts become indistinguishable and, thus, camera motion, i.e. visual odometry, cannot be directly obtained using these algorithms.

Alternative approaches exploit the spatial coherence of the scene flow by segmenting the range and/or color information and assuming a rigid motion for each of these segments [Gol+17; Jai+17; RA17]. Golyanik et al. [Gol+17] show that such scene flow methods outperform classical point-wise algorithms in terms of accuracy. Furthermore, assuming the largest cluster to be the static scene background, these methods enable visual odometry [Jai+17]. The major drawback of incorporating explicit segmentation into scene flow is the dependency between both, the scene flow and the segmentation. That is, per-image segmentation is lacking in temporal coherence and, thus cluster correspondences are difficult to establish. Furthermore, irregular or coarse segmentation results in false scene flow estimations while too fine a segmentation yields unstable and thus globally incoherent scene flow estimations.

In this chapter a novel scene flow estimation algorithm is proposed that *jointly* estimates the scene flow and enforces an implicit motion segmentation. This novel contribution facilitates two important improvements. Firstly, failure cases of previous methods, where the segmentation does not match the underlying scene flow, are eliminated. Segmentation and scene flow estimation are considered not as separate instances, but as directly interdependent. Secondly, scene flow clustering is incorporated into the model with negligible cost, since the presented method computes the clusters implicitly through regularization of the rigid-motion estimates via L_0 “norm”. This regularization leads to solutions containing unbiased, piecewise constant rigid-motion components. The primal dual hybrid gradient (PDHG) algorithm [CP11], is used along with the Moreau decomposition as discussed in section 3.1, to construct an efficient implementation that lends itself especially well to parallelization. This strategy essentially absorbs the unsupervised segmentation capabilities of the real-time Mumford-Shah approach [SC14] into a unified scene flow estimation.

This approach is evaluated on several real-world RGB-D image datasets, including the Bonn multi-body dataset [SB15], the TUM dataset [Stu+12], and the dataset provided by Jaimez et al. [Jai+17]. Photometric and geometric errors are computed as metrics on these realistic datasets. Furthermore, the endpoint-errors of static scenes are compared directly, using a reference scene flow, computed from ground truth camera poses. The results reveal a significant improvement of scene flow accuracy over existing approaches [Qui+14; Jai+15; Jai+17].

4.2 RELATED WORK

The estimation of the three-dimensional motion field, called scene flow, has its roots in the pioneering work of Vedula et al. [Ved+05]. Their work and ensuing publications had their focus on multi-view camera systems. However, this was changed by the emergence of affordable RGB-D cameras, able to measure both color and depth information simultaneously. Subsequently the focus will be on approaches that work on color and depth information and refer to the survey of Yan and Xiang [YX17] for other input types.

One of the first RGB-D scene flow estimations based on variational approaches, i.e. formulated as energy minimizations, was proposed by Herbst et al. [HRF13]. Their work shows how techniques from optical flow estimation such as the proposal of Brox et al. [Bro+04] can be transferred into the scene flow setting. They build a variational model consisting of linearized color consistency, depth consistency and a regularization term that penalizes the (smoothed) total variation of related points.

Quiroga et al. [QDC13] apply a combined local and global constraint in their variational scene flow estimation, where a set of 3D correspondences is used to deal with large displacements. They likewise use total variation (TV) to preserve motion discontinuities.

Jaimez et al. [Jai+15] apply a variational formulation including intensity and geometric consistency to RGB-D camera data. The main idea is to regularize the flow field with respect to the 3D geometry observed in the depth image instead of on the image plane using total variation, in order to achieve geometrically consistent results. Quiroga et al. [Qui+14] propose a scene flow reconstruction using local rigidity by modeling the 3D motion as a field of twists that consists of the three Euler rotation angles and a translation. This approach is difficult to steer, since translational and rotational motion can hardly be distinguished on a local level.

As scene flow estimation is an ill-posed problem, algorithms using a segmentation of the image plane have been developed in order to stabilize the estimation and further regularizing the per-segment flow.

Several approaches apply a *single, separate segmentation step*, which is especially helpful for online applications such as scene reconstruction [Zol+18]. Ghuffar et al. [Ghu+14] estimate the scene flow in a two-stage process using local scene flow estimation that integrates range flow and optical flow constraints into a global solution. They, furthermore, regularize the sum of differences of neighboring 3D velocities, which can be understood as an approximate total variation, and perform a graph-based motion segmentation in order to group trajectories based on depth and motion similarity.

Rünz and Agapito [RA17] propose a method that allows for the independent reconstruction of multiple rigidly moving objects. In their scene reconstruction approach, the current set of models is tracked for each new frame. The motion segmentation is formulated as a labeling problem using a fully connected Conditional Random Field, where the image is initially segmented into *simple linear iterative clustering (SLIC)* super-

pixels [Ach+12], using the geometric *iterative closest point* (ICP) cost [BM92; CM91] to assign a pixel to the rigid motion models as unary potentials.

Instead of separating segmentation and scene flow entirely, alternative approaches *alternate between scene segmentation and scene flow estimation*. Golyanik et al. [Gol+17] describe a multi-frame scene flow approach that models scene flow as a global non-linear least squares problem using an ICP-like formulation, whereas Jaimez et al. [Jai+17] perform a segmentation of the scene and classify the resulting motion clusters as static or moving elements, which allows for the separate estimation of the camera motion and the individual motions clusters observed in the scene. Initially, the scene is segmented using a K-means clustering on the depth channel, while the following optimization alternates between motion field estimation and smooth rigid segmentation.

Furthermore, a third class of methods model segmentation and motion estimation as a *joint variational approach*. Sun et al. [SSP15] estimate scene flow on the level of a finite number of depth layers extracted from the depth measurements. However, this yields a very strong constraint for the estimation of scene segmentation and flow. The initial segmentation is given by a K-means clustering of the depth values and the approach assumes a rigid motion for each depth layer. Innmann et al. [Inn+16] instead propose a method that reconstructs dynamic geometric shapes and operates on a fine volumetric grid that defines a piecewise rigid motion field. In this approach, geometry is implicitly modeled using truncated signed distance functions (TSDFs). In order to tackle the highly under-constrained non-rigid motion estimation, an *as rigid as possible* (ARAP) [SA07] regularization prior is applied on a regular volumetric grid, which is derived from the TSDF, which reads

$$\phi_R^{\text{ARAP}}(v, \mathcal{T}, \mathcal{R}) := \sum_{u \in \mathbb{A}(v)} \|(\mathcal{T}(v) - \mathcal{T}(u)) - \mathcal{R}(v) \cdot (\mathcal{C}(v) - \mathcal{C}(u))\|_2^2, \quad (4.1)$$

where v is a voxel (3D pixel) of the volumetric grid, $\mathbb{A}(v)$ is the set of voxels adjacent to v , voxels u are adjacent to v , $\mathcal{C} : \mathcal{V} \rightarrow \mathbb{R}^3$ is the canonical geometric model, $\mathcal{T} : \mathcal{V} \rightarrow \mathbb{R}^3$ is the estimated translation, and $\mathcal{R} : \mathcal{V} \rightarrow SO(3)$ are estimated rotation matrices. Similarly, Slavcheva et al. [Sla+17] address dynamic scenes reconstruction, using an implicit geometry-driven approach based on TSDFs. They utilize level set evolution without explicit correspondence search and estimate a dense, locally nearly isometric motion field using an approximate killing vector field (AKVF) [Sol+11] as regularization prior, which involves the Jacobian of the flow field.

In contrast to the existing scene flow estimation approaches discussed so far, the proposed technique in this chapter formulates rigid motion by regularizing the translational part $t \in \mathbb{R}^3$ and the rotational part $\alpha \in \mathbb{R}^3$ of the rigid motion. The implicit motion segmentation via the L_0 “norm” leads to a considerable stabilization of the flow estimation that cannot be reached by the previous techniques that considered the direct total variation, i.e. L_1 norm regularization of the scene flow gradient.

4.3 PROPOSED METHOD

The input to the proposed method is a sequence of RGB-D frames consisting of color (or grayscale) images $\mathcal{I}_i$ and depth maps $\mathcal{D}_i$. The corresponding color image and depth map are assumed to be temporally synchronized as well as spatially registered, i.e. the color $\mathcal{I}_i([p_x, p_y])$ corresponds to the depth value $\mathcal{D}_i([p_x, p_y])$ of the captured scene, for all pixels $[p_x, p_y] \in \Omega$ of the pixel set $\Omega \subset \mathbb{N}^2$. Furthermore, the frame rate of color images and depth maps is assumed to be sufficiently high ($\geq 10\text{Hz}$), such that the time difference between two consecutive color images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ and depth maps $\mathcal{D}_i$ and $\mathcal{D}_{i+1}$ is low.

Subsequently, let $C \in \mathbb{R}^{3 \times 3}$ be the camera matrix of the depth sensor, as defined in equation (2.7), and $\pi(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ be the camera projection, as defined in equation (2.8). Given a depth map $\mathcal{D}_i(\cdot)$, the corresponding point map $\mathcal{P}_i : \Omega \rightarrow \mathbb{R}^3$ is computed as in equation (2.13):

$$\mathcal{P}_i([p_x, p_y]) := C^{-1} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \cdot \mathcal{D}_i([p_x, p_y]) . \quad (4.2)$$

4.3.1 Scene Flow

The estimation of scene flow aims to compute the three-dimensional motions of observed scene points. To be more precise, for any pixel $p = [p_x, p_y] \in \Omega$, the goal is to estimate the function $s : \Omega \rightarrow \mathbb{R}^3$, which can be used to transform three-dimensional point $\mathcal{P}_i(p)$ to the corresponding point in the subsequent frame $\mathcal{P}_{i+1}(p')$, such that

$$\mathcal{P}_{i+1}(p') = \mathcal{P}_i(p) + s(p) \quad p, p' \in \Omega . \quad (4.3)$$

The corresponding pixel p' in $\mathcal{P}_{i+1}$ to pixel p in $\mathcal{P}_i$ is computed as follows

$$p' = \pi(\mathcal{P}_i(p) + s(p)) . \quad (4.4)$$

Equation (4.3) means that the scene flow $s(p)$ must lead to a consistent mapping between the consecutive point maps $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$. An illustration is shown in Fig. 4.1.

The derived equivalence in equation (4.3) is the central scene flow assumption and can also be understood as a form of *depth consistency assumption* [HRF13]. The term follows from the fact that the point maps are directly computed from the depth measurements, as shown in equation (4.2).

4.3.2 Rigid Scene Flow

Subsequently per-pixel motions in three-dimensional space are assumed to be represented as per-pixel rigid transformations via a *rotation map* $\mathcal{R} : \Omega \rightarrow SO(3)$ and a *translation map* $\mathcal{T} : \Omega \rightarrow \mathbb{R}^3$. This transformation field is referred to as *rigid scene flow*.

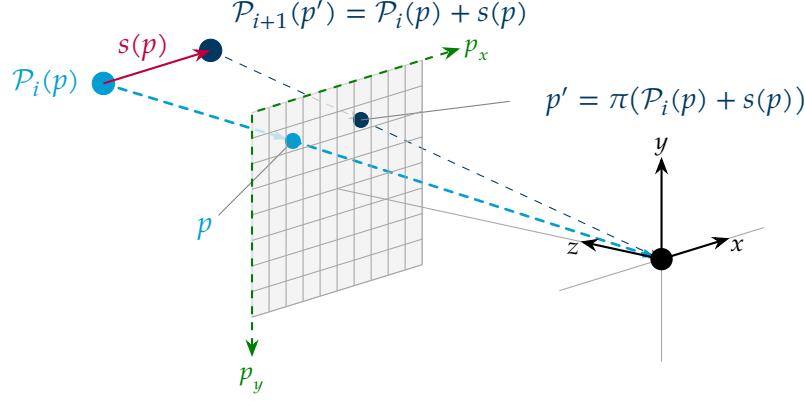


Figure 4.1: Illustration of the effect of the scene flow $s(p) \in \mathbb{R}^3$ on a 3D point $\mathcal{P}_i(p) \in \mathbb{R}^3$ and its new value $\mathcal{P}_{i+1}(p') \in \mathbb{R}^3$ and pixel position $p' \in \Omega$ in the next RGB-D frame.

Similar to equation (4.3) one can derive the relation between two consecutive point maps $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ as

$$\mathcal{P}_{i+1}(p') = \mathcal{R}(p)\mathcal{P}_i(p) + \mathcal{T}(p) \quad p, p' \in \Omega . \quad (4.5)$$

The corresponding pixel p' is

$$p' = \pi(\mathcal{R}(p)\mathcal{P}_i(p) + \mathcal{T}(p)) . \quad (4.6)$$

The relation between *scene flow* and *rigid scene flow* can be derived by equating the right-hand sides of equation (4.3) and equation (4.5), which leads to

$$s(p) = (\mathcal{R}(p) - I_{3 \times 3}) \mathcal{P}_i(p) + \mathcal{T}(p) , \quad (4.7)$$

where $I_{3 \times 3} \in \mathbb{R}^{3 \times 3}$ is the 3x3 identity matrix.

Assuming that two neighboring pixels $p, q \in \Omega$ undergo the same rigid transformations, i.e. $\mathcal{R}(p) = \mathcal{R}(q)$ and $\mathcal{T}(p) = \mathcal{T}(q)$, from equation (4.7) then follows

$$\|s(p) - s(q)\|_2^2 = \|(\mathcal{R}(p) - I_{3 \times 3}) (\mathcal{P}_i(p) - \mathcal{P}_i(q))\|_2^2 . \quad (4.8)$$

The sum of the left-hand side of equation (4.8) over all neighbors q of p , corresponds to $\|\nabla s(p)\|_2^2$. This means that the regularization $\Phi_R(\nabla s(p)) = \|\nabla s(p)\|_2^2$ would favor solutions $s(p) \approx s(q)$, which in turn leads to $\mathcal{R}(p) \approx I_{3 \times 3}$ on the right-hand side of equation (4.8). This means that variational regularization on the scene flow, favors solutions with vanishing rotations. Thus, intuitively, a regularization of *rigid* scene flow parameters should improve scene flow estimation results in the case of rotational motions.

4.3.3 Linearized Rigid Scene Flow

Due to the made assumption of dense temporal acquisition of the RGB-D data stream, rigid motions of points between two consecutive point maps $\mathcal{P}_i$ and $\mathcal{P}_{i+1}$ can be assumed to be *small*. From the *Rodrigues'* rotation formula [Rod40]

$$R = I_{3 \times 3} + \sin(\theta) [\mathbf{r}]_{\times} + (1 - \cos(\theta)) [\mathbf{r}]_{\times}^2, \quad (4.9)$$

and the *small-angle approximations* of sine and cosine functions

$$\sin(\theta) \approx \theta \quad (4.10)$$

$$\cos(\theta) \approx 1 \quad (4.11)$$

$$(4.12)$$

follows that a rotation matrix $R \in SO(3)$ for a small rotation can be rewritten as

$$R \approx I_{3 \times 3} + [\boldsymbol{\alpha}]_{\times}. \quad (4.13)$$

The term $[\cdot]_{\times}$ denotes the *skew-symmetric cross-product* matrix

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{\times} := \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix}, \quad (4.14)$$

with $\boldsymbol{\alpha} \in \mathbb{R}^3$, rotation angle $\theta = \|\boldsymbol{\alpha}\|_2$, and rotation-axis $\mathbf{r} = \frac{\boldsymbol{\alpha}}{\|\boldsymbol{\alpha}\|_2} \in \mathbb{S}^2 = \{\mathbf{x} \in \mathbb{R}^3 \mid \|\mathbf{x}\|_2 = 1\}$.

Using the derived approximation for small rotations and the anti-commutativity property of the cross product, i.e. $[\mathbf{x}]_{\times} \cdot \mathbf{y} = -[\mathbf{y}]_{\times} \cdot \mathbf{x}$, equation (4.7) can be linearized as

$$s(p) = -[\mathcal{P}_i(p)]_{\times} \cdot \mathcal{A}(p) + \mathcal{T}(p), \quad (4.15)$$

with $\mathcal{A} : \Omega \rightarrow \mathbb{R}^3$ and $\mathcal{T} : \Omega \rightarrow \mathbb{R}^3$ being the rotational and translational parts of a rigid scene flow estimation respectively. Subsequently, this linearized version will be referred to as *rigid scene flow*.

4.3.4 Optimization Problem Formulation

Describing the scene flow as shown in equation (4.15), allows to formulate the problem of scene flow estimation as an energy minimization over the two maps $\mathcal{A}, \mathcal{T} : \Omega \rightarrow \mathbb{R}^3$ as follows

$$E_{\text{RSF}}(\mathcal{A}, \mathcal{T}) = \sum_{p \in \Omega} \phi_{\text{D}}(\mathcal{A}, \mathcal{T}, p) + \phi_{\text{R}}(\nabla \mathcal{A}(p), \nabla \mathcal{T}(p)). \quad (4.16)$$

The first part of this energy, the data term ϕ_{D} , penalizes deviations from the rigid scene flow characterizations of the input data. The second term ϕ_{R} is a regularizer, that penalizes globally inconsistent solutions based on prior knowledge about the desired solutions.

Data Term

The data term ϕ_D consists of two components ϕ_P and ϕ_I , which describe the *depth consistency assumption* related to the point maps $\mathcal{P}_i$ and the *brightness consistency assumption* with respect to the color images $\mathcal{I}_i$.

The former premise of the data term is derived via the linearization of the rigid scene flow in equation (4.15) as

$$\mathcal{P}_{i+1}(p') - \mathcal{P}_i(p) = -[\mathcal{P}_i(p)]_{\times} \cdot \mathcal{A}(p) + \mathcal{T}(p) . \quad (4.17)$$

Since the corresponding pixel p' depends on the rigid scene flow of pixel p (cf. equation (4.6)), subsequently it will be written as a function $p' : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \Omega$

$$p'(\mathcal{A}(p), \mathcal{T}(p)) = \pi(\mathcal{P}_i(p) - [\mathcal{P}_i(p)]_{\times} \cdot \mathcal{A}(p) + \mathcal{T}(p)) . \quad (4.18)$$

This leads to the definition of ϕ_P as

$$\phi_P(\mathcal{A}, \mathcal{T}, p) = \|\mathcal{P}_{i+1}(p'(\mathcal{A}(p), \mathcal{T}(p))) - \mathcal{P}_i(p) + [\mathcal{P}_i(p)]_{\times} \cdot \mathcal{A}(p) - \mathcal{T}(p)\|_1 . \quad (4.19)$$

This component of the energy functional can be understood as a form of the *depth consistency assumption*.

The latter premise of the data term is that colors do not change between two consecutive images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ for corresponding pixels, i.e.

$$\mathcal{I}_{i+1}(p') = \mathcal{I}_i(p) . \quad (4.20)$$

This so-called *brightness consistency assumption* leads to the definition of ϕ_I

$$\phi_I(\mathcal{A}, \mathcal{T}, p) = \|\mathcal{I}_{i+1}(p'(\mathcal{A}(p), \mathcal{T}(p))) - \mathcal{I}_i(p)\|_1 . \quad (4.21)$$

The overall data term is then

$$\phi_D(\mathcal{A}, \mathcal{T}, p) = \phi_I(\mathcal{A}, \mathcal{T}, p) + \lambda_P \phi_P(\mathcal{A}, \mathcal{T}, p) . \quad (4.22)$$

The non-negative parameter λ_P steers the impact of the components of the data term.

Regularization Term

Solely optimizing for the data term described above would lead to incoherent and noisy point-wise solutions. Therefore, regularization terms are indispensable to construct plausible solutions. This is achieved by regularizing piecewise rigid motion regularization over the variation of the rotational and translational parts, i.e. $\nabla \mathcal{A}(p)$ and $\nabla \mathcal{T}(p)$ respectively, with the L_0 “norm”

$$\phi_R(\mathcal{A}(p), \mathcal{T}(p)) = \phi_A(\mathcal{A}(p)) + \phi_T(\mathcal{T}(p)) \quad (4.23)$$

$$= \|\nabla \mathcal{A}(p)\|_0 + \|\nabla \mathcal{T}(p)\|_0 . \quad (4.24)$$

This regularization is equal to the *Pott's model* introduced in chapter 3. It is important to note that this form of the L_0 “norm” regularization, can not be applied to the scene flow s directly, as it would favor an unrealistic, piecewise constant motions. In contrast, the proposed regularization of $\mathcal{A}$ and $\mathcal{T}$ favors *piecewise rigid* motions.

4.3.5 Optimization Problem Linearization

The two components $\phi_{\mathcal{I}}$ and $\phi_{\mathcal{T}}$ in the data term $\phi_{\mathcal{D}}$ are nonlinear and non-convex, prohibiting efficient convex optimization algorithms. In order to linearize the nonlinear terms, the well-known strategy of *successive linearizations* and *warpings*, proposed by Brox et al. [Bro+04], is utilized. The idea behind this approach is sketched in Fig. 4.2. First the input images are down-sampled to smaller resolutions, at which the nonlinear parts of the data term $\phi_{\mathcal{D}}$ can be linearly approximated with sufficient precision. The linearized optimization problem is then solved on the coarsest resolution. The result is up-sampled to the next higher resolution and is used as initial values (kickstart). This process is repeated up to the original resolution.

Subsequently the approach for one single level is derived. Let $\mathbf{a} = \mathcal{A}(p)$ and $\mathbf{t} = \mathcal{T}(p)$, then the linearization of $\phi_{\mathcal{I}}$ and $\phi_{\mathcal{T}}$ at initial rigid scene flow values $\mathbf{a}_0, \mathbf{t}_0$ is achieved by using first-order Taylor series expansion on $\mathcal{M} \in \{\mathcal{I}, \mathcal{P}\}$ as follows:

$$\begin{aligned} \mathcal{M}_{i+1}(p'(\mathbf{a}, \mathbf{t})) &\approx \mathcal{M}_{i+1}(p'(\mathbf{a}_0, \mathbf{t}_0)) \\ &+ \nabla \mathcal{M}_{i+1}(p'(\mathbf{a}_0, \mathbf{t}_0)) \cdot \nabla p'(\mathbf{a}_0, \mathbf{t}_0) \cdot ((\mathbf{a}, \mathbf{t})^\top - (\mathbf{a}_0, \mathbf{t}_0)^\top) . \end{aligned} \quad (4.25)$$

The values of $\mathbf{a}_0$ and $\mathbf{t}_0$ are equal to the up-scaled results of the previous (coarser) level, and are equal to $\mathbf{0}$ for the coarsest level.

The corresponding pixel p' is computed as in equation (4.18), and its gradient is

$$\nabla p'(\mathbf{a}, \mathbf{t}) = \nabla \pi(p'(\mathbf{a}, \mathbf{t})) \cdot \mathbf{C} \cdot [-\mathcal{P}_i(p) \mid I_{3 \times 3}] . \quad (4.26)$$

Applying this linearization to equation (4.19) and equation (4.21) leads to

$$\phi_{\mathcal{I}}(\mathbf{a}, \mathbf{t}) = \|A_{\mathcal{I}}(\mathbf{a}, \mathbf{t})^\top - \mathbf{b}_{\mathcal{I}}\|_1 \quad (4.27)$$

$$\phi_{\mathcal{P}}(\mathbf{a}, \mathbf{t}) = \|A_{\mathcal{P}}(\mathbf{a}, \mathbf{t})^\top - \mathbf{b}_{\mathcal{P}}\|_1 \quad (4.28)$$

where

$$A_{\mathcal{I}} = \nabla \mathcal{I}(p'(\mathbf{a}_0, \mathbf{t}_0)) \cdot \nabla p'(\mathbf{a}_0, \mathbf{t}_0) \quad (4.29)$$

$$A_{\mathcal{P}} = \nabla \mathcal{P}(p'(\mathbf{a}_0, \mathbf{t}_0)) \cdot \nabla p'(\mathbf{a}_0, \mathbf{t}_0) + [\mathcal{P}_i \mid -I_{3 \times 3}] , \quad (4.30)$$

and

$$\mathbf{b}_{\mathcal{I}} = \mathcal{I}_i - \mathcal{I}_{i+1} + A_{\mathcal{I}} \cdot [\mathbf{a}_0, \mathbf{t}_0]^\top \quad (4.31)$$

$$\mathbf{b}_{\mathcal{P}} = \mathcal{P}_i - \mathcal{P}_{i+1} + A_{\mathcal{P}} \cdot [\mathbf{a}_0, \mathbf{t}_0]^\top . \quad (4.32)$$

Since the L_1 norm is used in the entire data term, the overall linearized data term ϕ_{Dlin} can be written as

$$\phi_{\text{Dlin}}(\mathbf{a}, \mathbf{t}) = \left\| A_{\text{Dlin}} \begin{bmatrix} \mathbf{a} \\ \mathbf{t} \end{bmatrix} - \mathbf{b}_{\text{Dlin}} \right\|_1 \quad (4.33)$$

$$= \phi_{L_1} \left(A_{\text{Dlin}} \begin{bmatrix} \mathbf{a} \\ \mathbf{t} \end{bmatrix} - \mathbf{b}_{\text{Dlin}} \right) , \quad (4.34)$$

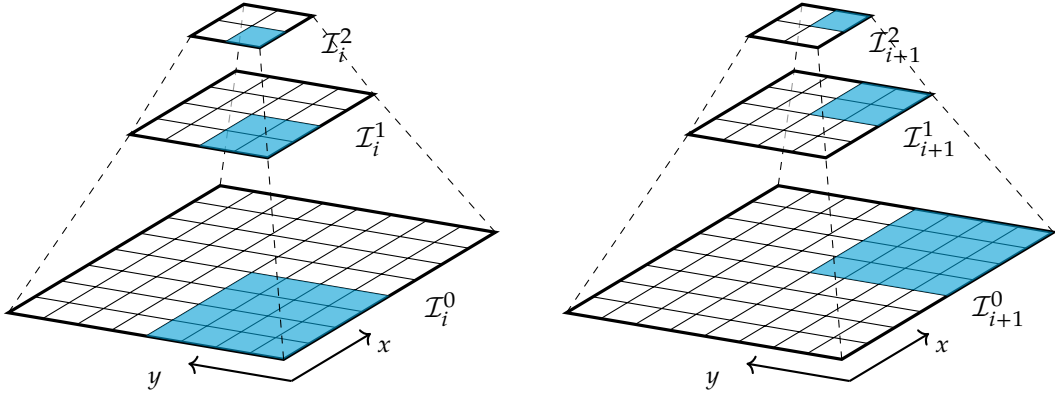


Figure 4.2: This figure sketches the idea behind *successive linearizations and warps* [Bro+04]. Images $\mathcal{I}_i$ and $\mathcal{I}_{i+1}$ are subsampled to lower resolutions for various levels, denoted as $\mathcal{I}_i^l, \mathcal{I}_{i+1}^l$, with $l = 0$ being the original resolution. It can be observed that a motion of 4 pixels between $\mathcal{I}_i^0$ and $\mathcal{I}_{i+1}^0$ (blue pixels) becomes a motion of 2 pixels between $\mathcal{I}_i^1$ and $\mathcal{I}_{i+1}^1$, and a motion of only 1 pixel between $\mathcal{I}_i^2$ and $\mathcal{I}_{i+1}^2$. The idea is to estimate motions on the coarsest level and use the result as an initialization (*kickstart*) for the next finer level.

with $A_{Dlin} = [A_{\mathcal{I}}^T | \lambda_{\mathcal{P}} A_{\mathcal{P}}^T]^T$ and $\mathbf{b}_{Dlin} = [\mathbf{b}_{\mathcal{I}}^T \quad \lambda_{\mathcal{P}} \mathbf{b}_{\mathcal{P}}^T]^T$.

Similar to ϕ_{Dlin} the regularization term ϕ_R is rewritten as

$$\phi_R(\boldsymbol{\alpha}, \mathbf{t}) = \left\| A_R \cdot \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{t} \end{bmatrix} \right\|_0 \quad (4.35)$$

$$= \phi_{L_0} \left(A_R \cdot \begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{t} \end{bmatrix} \right) \quad (4.36)$$

with

$$A_R = \begin{bmatrix} \lambda_{dt} \nabla & \mathbf{0} \\ \mathbf{0} & \lambda_{d\alpha} \nabla \end{bmatrix}. \quad (4.37)$$

The overall linearized optimization problem reads

$$E_{RSF}(\mathcal{A}, \mathcal{T}) = \sum_{p \in \Omega} \phi_{Dlin}(\mathcal{A}(p), \mathcal{T}(p)) + \phi_R(\mathcal{A}(p), \mathcal{T}(p)) \quad (4.38)$$

$$= \sum_{p \in \Omega} \phi_{L_1} \left(A_{Dlin} \begin{bmatrix} \mathcal{A}(p) \\ \mathcal{T}(p) \end{bmatrix} - \mathbf{b}_{Dlin} \right) + \phi_{L_0} \left(A_R \cdot \begin{bmatrix} \mathcal{A}(p) \\ \mathcal{T}(p) \end{bmatrix} \right). \quad (4.39)$$

4.3.6 Primal-Dual Formulation on Finite Weighted Graphs

In this section the previously derived energy formulation of rigid scene flow estimation is defined on finite weighted graphs (cf. section 2.3) and solved via PDHG (cf. sections 2.4.3 and 2.4.4). This is accomplished by creating graphs G_i^l and G_{i+1}^l for every down-sampled image level $l \in \{0, \dots, L\}$, as discussed in section 4.3.5 and illustrated in Fig. 4.2. The vertices are connected in a *2-neighborhood*, such that ∇_w (cf. equation (2.48)) corresponds to horizontal and vertical forward-differences.

For each down-sampled level l , as discussed in section 2.4.5, the relation to PDHG is established through the following definitions. The (unknown) vector $\mathbf{f}$ corresponds to the piecewise constant function $f_{\mathbb{P}}(\cdot)$ stacked on top of each other for all $v \in \mathcal{V}$, similar to equation (2.112)

$$\mathbf{f} = \begin{bmatrix} f_{\mathbb{P}}(v_1) \\ \vdots \\ f_{\mathbb{P}}(v_{|\mathcal{V}|}) \end{bmatrix}. \quad (4.40)$$

The piecewise constant function $f_{\mathbb{P}}(v)$ is equal to the estimated (piecewise constant) maps $\mathcal{A}, \mathcal{T}$

$$f_{\mathbb{P}}(v) = \begin{bmatrix} \mathcal{A}(v) \\ \mathcal{T}(v) \end{bmatrix}. \quad (4.41)$$

The energy minimization problem in equation (4.39) can then be rewritten in terms of PDHG as

$$G(\mathbf{f}) = 0 \quad (4.42)$$

$$F(K\mathbf{f}) = \sum_{v \in \mathcal{V}} \underbrace{\|A_{Dlin} \cdot f_{\mathbb{P}}(v) - \mathbf{b}_{Dlin}\|_1 + \|A_R \cdot f_{\mathbb{P}}(v)\|_0}_{\phi_F} \quad (4.43)$$

$$K_v = \begin{bmatrix} A_{Dlin} \\ A_R \end{bmatrix}. \quad (4.44)$$

To address the computational challenges associated with solving L_0 regularized problems, the truncated quadratic “norm” regularization $\phi_{\alpha TQ}$ is employed as an alternative. As discussed in section 3.1, the convex conjugate of $\phi_{\alpha TQ}$ is zero, which causes the optimization process to neglect the regularization term, potentially yielding suboptimal results. To mitigate this issue, *Moreau’s identity* (cf. equation (2.88)) is applied to approximate $\text{prox}_{\sigma F^*}(\sigma \mathbf{y})$, as follows

$$\text{prox}_{\sigma \phi_F^*}(\sigma \mathbf{y}) = \mathbf{y} - \text{prox}_{\frac{1}{\sigma} \phi_F}(\mathbf{y}). \quad (4.45)$$

This transforms the primal-dual structure into a primal-proximal algorithm, whose convergence properties are closely connected to non-convex ADMM approaches [WYZ19].

Prior to deriving the proximal mapping of $\phi_F(K_v f_{\mathbb{P}}(v))$, it is worth to note that this term can be thought of as a two-parameter function

$$\phi_F(A_{Dlin} f_{\mathbb{P}}(v), A_R f_{\mathbb{P}}(v)) = \phi_{L_1}(A_{Dlin} f_{\mathbb{P}}(v)) + \phi_{\alpha TQ}(A_R f_{\mathbb{P}}(v)). \quad (4.46)$$

Hence the separable sum property of the proximity operator (cf. equation (2.89)) can be applied, to compute $\text{prox}_{\frac{1}{\sigma}\phi_F}(\mathbf{y}_1, \mathbf{y}_2)$

$$\text{prox}_{\frac{1}{\sigma}\phi_F}(\mathbf{y}_1, \mathbf{y}_2) = \begin{bmatrix} \text{prox}_{\frac{1}{\sigma}\phi_{L_1}}(\mathbf{y}_1) \\ \text{prox}_{\frac{1}{\sigma}\phi_{\alpha TQ}}(\mathbf{y}_2) \end{bmatrix}. \quad (4.47)$$

To obtain piecewise constant results, the limit case $\alpha \rightarrow +\infty$ must be considered, which overall leads to (cf. equations (3.20) and (3.22)):

$$\text{prox}_{\frac{1}{\sigma}\phi_{L_1}}(\mathbf{y}) = \text{sign}(\mathbf{y} - \mathbf{b}_{Dlin}) \max\left(|\mathbf{y}_1| - \frac{1}{\sigma}, \mathbf{b}_{Dlin}\right) \quad (4.48)$$

$$\lim_{\alpha \rightarrow +\infty} \text{prox}_{\sigma\phi_{\alpha TQ}}(\mathbf{y}) = \begin{cases} \mathbf{0} & \text{if } \|\mathbf{y}\|_2^2 \leq 2\lambda\sigma \\ \mathbf{y} & \text{otherwise} \end{cases} \quad (4.49)$$

The remaining term $\text{prox}_{\tau_G}(\cdot)$ follows from equation (2.94)

$$\text{prox}_{\tau_G}(\mathbf{x}) = \mathbf{x}. \quad (4.50)$$

The entire problem is solved as shown in algorithm 4.1, where PPDHG_Alg2 is Alg2 of the *Preconditioned Primal-Dual Hybrid Gradient* method, i.e. algorithm 2.4 equipped with preconditioning as described in section 2.4.4.

Algorithm 4.1 L_0 Rigid Scene Flow Estimation

- 1: Input: $\mathcal{I}_i, \mathcal{D}_i, \mathcal{I}_{i+1}, \mathcal{D}_{i+1}$
 - 2: $\mathcal{I}_i^l, \mathcal{D}_i^l, \mathcal{I}_{i+1}^l, \mathcal{D}_{i+1}^l \leftarrow$ subsample inputs (cf. section 4.3.5 and Fig. 4.2)
 - 3: $\mathcal{A}_0^N, \mathcal{T}_0^N \leftarrow \mathbf{0}$
 - 4: **for** $l = L \dots 0$ **do**
 - 5: $G_i^l, G_{i+1}^l \leftarrow$ create graphs from input images $\mathcal{I}_i, \mathcal{D}_i$ and $\mathcal{I}_{i+1}, \mathcal{D}_{i+1}$ respectively, via two-neighborhood (forward differences)
 - 6: $\mathbf{a}_0, \mathbf{t}_0 \leftarrow \mathcal{A}_0^l, \mathcal{T}_0^l$
 - 7: $A_{\mathcal{I}}, A_{\mathcal{P}}, b_{\mathcal{I}}, b_{\mathcal{P}} \leftarrow$ as in equations (4.29) to (4.32)
 - 8: $\mathcal{A}^l, \mathcal{T}^l \leftarrow$ PPDHG_Alg2 with $\text{prox}_{F^*}, \text{prox}_G$ as in equations (4.45) and (4.50)
 - 9: **if** $l > 0$ **then**
 - 10: $\mathcal{A}_0^{l-1}, \mathcal{T}_0^{l-1} \leftarrow$ upscale $\mathcal{A}^l, \mathcal{T}^l$ (cf. section 4.3.5 and Fig. 4.2)
 - 11: **end if**
 - 12: **end for**
 - 13: Output: $\mathcal{A}, \mathcal{T} \leftarrow \mathcal{A}^0, \mathcal{T}^0$
-

This demonstrates the great attractiveness of the primal dual algorithm. The intricate non-convex energy constructed in equation (4.16) in section 4.3.4 can be rewritten as a series of linear operators and simple, point-wise nonlinear operations. This allows for an efficient implementation that lends itself well to pixel-level parallel executions interlaced with well-developed linear algebra calls.

4.4 EVALUATION

In this section the previously derived approach is evaluated and compared against other approaches. For comparison and evaluation, RGB-D image pairs from three different real world datasets are used: The Bonn multi-body benchmark [SB15], the TUM benchmark [Stu+12], and the supplemented datasets from Jaimez et al. [Jai+17]. Table 4.1 gives an overview over the used image pairs and datasets.

ID	Full Name	Dyn	Bench.	Frames
bonn-tc	Tea can	y	Bonn	700-701
fr1-desk	freiburg1_desk	n	TUM	200-201
fr1-xyz	freiburg1_xyz	n	TUM	203-204
fr3-wh	freiburg3_Walk_Halfsphere	y	TUM	1-2
vosf-rob	Robot	y	[Jai+17]	1-2
vosf-cw	Cleaning whiteborad	y	[Jai+17]	1-2

Table 4.1: RGB-D image pairs used for evaluation: Abbreviation used in the paper (**ID**), full name of dataset (**Full Name**), dynamic scene (**Dyn**), benchmark (**Bench.**), and frames used (**Frames**).

For evaluation the proposed method (L_0) is compared to the following approaches

- *Semi-Rigid Scene Flow (SRSF)* by Quiroga et al. [Qui+14], which utilizes a L_1 regularization on the rigid scene flow.
- *Primal-Dual Flow (PDFlow)* by Jaimez et al. [Jai+15], which utilizes a L_1 regularization on the scene flow.
- *Visual Odometry Scene Flow (VOSF)* by Jaimez et al. [Jai+17], which utilizes k-Means [Llo82] to cluster the color image and then estimates rigid transformation per cluster.

The presented approach was implemented in the *PROST* framework [Möl+16], which was designed for solving large-scale problems with proximal structure. This general purpose optimization framework however, does not allow for a real-time implementation. It is worth to note that the complexity of the optimization algorithm is effectively equivalent to the **PDFlow** approach of Jaimez et al. [Jai+15], which runs in real-time.

As shown in the previous section, the behavior of this algorithm can be controlled through a set of parameters. Throughout all experiments, these parameters were set to $\lambda_P = 2.0, \lambda_{dt} = 2.0, \lambda_{d\alpha} = 0.5, \tau_0 = 0.25, \sigma_0 = 1.5$, which yields a good trade-off between robustness and accuracy.

As discussed in section 3.1, the result is neither piecewise constant nor does it contain partitions. However, the segmentation can be computed from the final rigid motion parameters α and $\mathbf{t}$ via graph cuts, k-means, or by thresholding the normalized L_2 norm of these values. Fig. 4.5 shows that these implicitly computed motion clusters are competitive to explicitly computed segmentations.

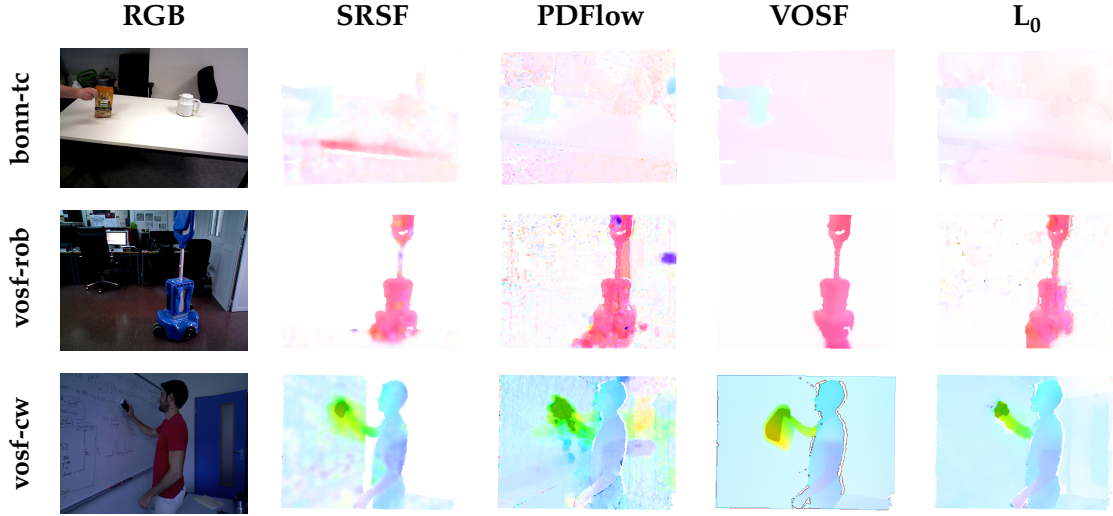


Figure 4.3: Color-coded projections of the scene flow estimates of **SRSF** [Qui+14], **PDFlow** [Jai+15], **VOSF** [Jai+17] and the proposed L_0 approach on dynamic scenes. Note how the L_0 approach combines both, robustness and accuracy. Consider especially the right arm of the left person in **tum fr3-wh** or the white board eraser in **vosf-cw**.

4.4.1 Evaluation without Reference Flows

A visual comparison of the previously mentioned approaches shown in Fig. 4.3. A color-coded projection of the three-dimensional scene flow estimations onto the image plane is shown, which demonstrates the accuracy and robustness of the proposed approach on real-world data. Compared to **SRSF** and **PDFlow**, the proposed scene flow estimation contains significantly less noise and the moving objects in the scene are consistently estimated. Compared to the proposed approach, **VOSF** produces visually smoother scene flows. However, incorporating the photometric and geometric error measures of valid pixels shown in Table 4.2, yields that the actual accuracy of the proposed approach is significantly better. The indirect regularization of the scene flow via the implicit L_0 “norm” segmentation of the rigid motion components avoids over-smoothing in contrast to the direct regularization of the **VOSF** approach.

ID	metric	SRSF	PDFlow	VOSF	L_0
bonn-tc	PE	0.032	0.031	0.033	0.027
	GE	0.023	0.029	0.033	0.018
vosf-rob	PE	0.054	0.067	0.050	0.040
	GE	0.130	0.146	0.127	0.093
vosf-cw	PE	0.048	0.037	0.041	0.029
	GE	0.068	0.068	0.074	0.045

Table 4.2: Photometric errors (PE) and geometric errors (GE) for the datasets shown in Fig. 4.3.

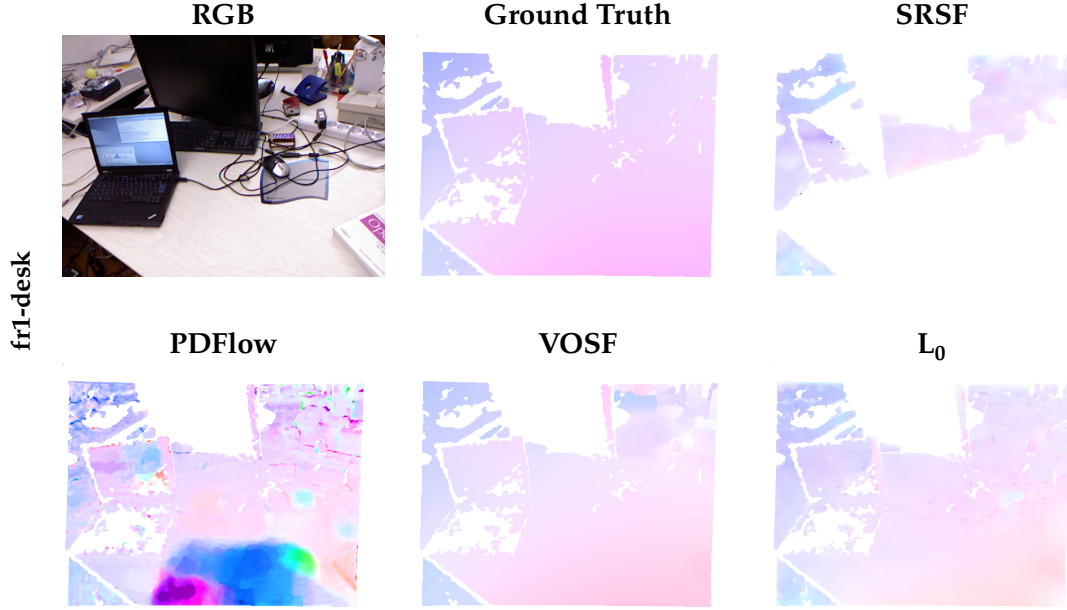


Figure 4.4: Projection of the scene flow estimates of **SRSF** [Qui+14], **PDFlow** [Jai+15], **VOSF** [Jai+17] and the proposed **L₀** approach for static scenes with reference scene flow from the TUM dataset [Stu+12]. The reference scene flow was estimated from ground truth camera pose. Note how the smoothing effect of the **VOSF** approach is useful in this mostly planar scene.

4.4.2 Evaluation with Reference Flows

The static scenes in Fig. 4.4 allow for an estimation of the reference scene flow from the given ground truth camera pose. This allows for a comparison of the endpoint error (EPE), i.e. the L_2 norm of the differences of ground truth and estimated scene flow. The EPEs of these two scenes are shown in Table 4.3, while the color-coded projections of the scene flows are shown in Fig. 4.4. It turns out that the proposed approach leads to a lower EPE. This is due to the fact that the static scene can be described by a single rigid motion, the motion of the camera. The L_0 approach is well-suited for this kind of scene flow estimation, as it directly optimizes for rigid motions. This effect is visualized in Fig. 4.5, where it can be observed that the proposed approach leads to a single consistent motion cluster.

ID	metric	SRSF	PDFlow	VOSF	L0
fr1-desk	PE	0.112	0.082	0.070	0.048
	GE	0.087	0.117	0.115	0.065
	EPE	5.332	7.132	4.385	4.236

Table 4.3: Photometric errors (PE), geometric errors (GE) and end-point errors (EPE) for the datasets shown in Fig. 4.4.

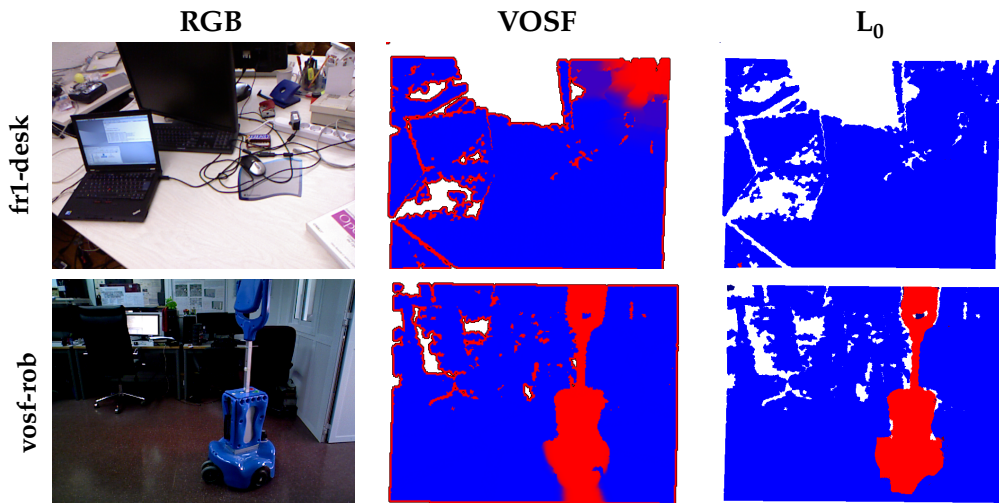


Figure 4.5: Visual comparison of motion segments for **VOSF** and L_0 approach using a threshold of 0.86 on the normalized L_2 norm of the vector $[\alpha, \mathbf{t}]$. The top example shows the static scene **fr1-desk**, i.e. a scene with one single motion cluster. This is well approximated by both, although **VOSF** is not as robust in the upper right corner. The bottom example shows a dynamic scene, where both approaches achieve comparable results.

4.5 CONCLUSION

A model of piece-wise rigid scene flow was formulated via L_0 regularization of the rotational and translational motion components. This formulation strongly contributes to the improvements in terms of accuracy of the scene flow estimation.

Failure cases of the proposed model are RGB-D frames, where the rigid-motion approximation in (4.13) is violated by rapid rotations. Such cases still produce a globally reasonable scene flow, but the motion of these objects is not recovered. This can be observed for the dataset **vosf-cw** in Fig. 4.3. Here, the fast moving white board eraser is not precisely reconstructed by the proposed L_0 approach, while the other parts of the scene and especially the background motion are accurately estimated. Yet compared to previous methods the error in the vicinity of the eraser is considerably reduced.

Further failures can occur from strongly non-rigid motions, such as elasto-plastic deformations. Here, the deformations will be approximated by piecewise-rigid transformations, leading to incorrect results.

The implicit motion segmentation presented in this work is capable of stabilizing variational scene flow estimation, resulting in piecewise-rigid flows. In contrast to other methods that rely on segmentation, the proposed approach utilizes the clustering properties of the L_0 regularization and thus performs the clustering of motions implicitly.

COHERENT DENOISING OF DEPTH IMAGES AND NORMAL MAPS

In this chapter, a versatile non-planar model approach for coherent enhancement of depth and normal maps is discussed. It is specifically designed to handle both planar and non-planar geometry. To achieve this, the underlying problem is formulated as a partitioning of non-planar model parameters on weighted finite graphs. Solving this problem involves utilizing a modified version of the Cut Pursuit algorithm, introduced in section 3.2. This algorithm efficiently divides input data into regions of similar geometric models. These regions and their associated model parameters are leveraged to compute a denoised and infilled depth image, along with its coherent normal map.

This chapter begins with a motivation behind the denoising of depth images and highlights the problem of denoising depth and normal maps independently. Following this, a review of related work is presented. Next, details of the proposed approach are discussed. Finally, the proposed approach is evaluated, and the results are discussed to highlight its effectiveness.

The approach discussed in this chapter was published [GMK24] and presented at the International Conference on 3D Vision (3DV) 2024 in Davos, Switzerland.

5.1 MOTIVATION

Accurate geometric information is crucial for a wide range of applications in image processing and computer vision, including geometric reconstruction [Zol+18] and visual odometry [PG18]. However, depth data acquired from sensors inevitably contains noise and missing values due to physical reasons, such as the inherent spatial and temporal limitations of sensing technologies. The severity of these disturbances is influenced by factors such as the distance to the object and coordinates of the pixel, as discussed by Sarbolandi et al. [SLK15]. As a consequence, these effects can substantially undermine the performance of algorithms that heavily rely on dependable geometric information, potentially leading to inaccurate results or even failure.

To mitigate this issue, it is essential for those algorithms to apply pre-processing techniques, such as inpainting missing values and denoising depth images and normal maps. For simplicity, many current approaches perform pre-processing that address the

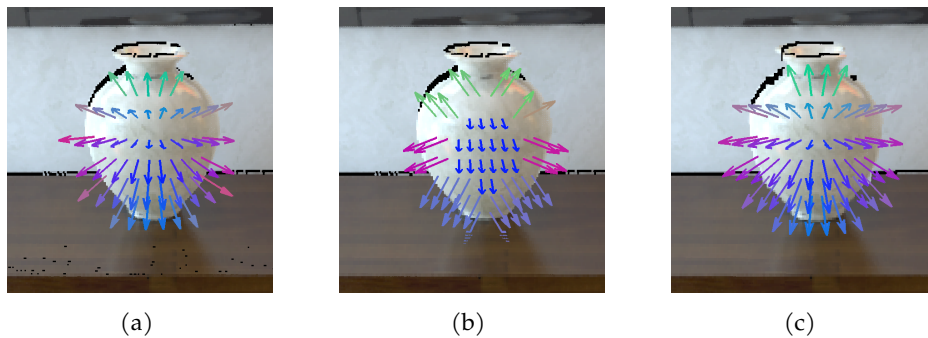


Figure 5.1: This figure shows a colored point cloud along with surface normals of the living room scene from ICL NUIM dataset [Han+14]. Subfigures (a)-(c) show the ground truth, a piecewise planar estimation, and the result of the proposed approach, respectively. Note how the piecewise planar solution exhibits a truncated icosahedron-like shape, with piecewise constant normals. On the other hand, the proposed approach is better suited to handle this rather complex shape and yields normals that are close to the ground truth.

problem of noise and missing values in depth images and normal maps. For simplicity, many current approaches perform pre-processing of depths and normals independently [Zol+18].

This simple yet effective procedure can lead to incoherent and inaccurate geometric information, i.e. the denoised normals may not correspond to the surface information in the denoised depth image. To overcome this problem, it is possible to leverage approaches which focus on fitting of simple geometric models, such as planes [FTK14; PG18; RMB21] or cylinders [PG18]. These models can then be used to compute denoised depth images and normal maps that are coherent with each other.

The estimation of simple geometric models for denoising, however, is not well-suited for scenes featuring objects with sufficiently complex geometry, such as spheres, as shown in Fig. 5.1. The planar model [RMB21] in Fig. 5.1(b) is insufficient to represent a non-planar geometry, and this limitation necessitates advanced methods to handle more complex shapes while ensuring consistency between depth and normals (cf. Fig. 5.1(c)).

In this chapter, a novel non-planar model approach is introduced, which addresses the problem of denoising depth images and normal maps of simple as well as complex geometries coherently, with high accuracy. The key to this method lies in the representation of inverse depth as a polynomial of second-order. The underlying mathematical problem is formalized as a partitioning of finite weighted graphs, and solved via a custom version of the Cut Pursuit algorithm, introduced in section 3.2. The performance of this approach is analyzed on synthetic as well as real-world datasets, both quantitatively and qualitatively. Unlike previous literature [Kor+22], which focused on quality of clustering, this chapter focuses on the accuracy of the estimated models.

5.2 RELATED WORK

Image denoising is a fundamental task in image processing that addresses the inherent noise present in camera sensors since their early days. This noise can have a significant impact on the overall quality of images and affect various applications. To avoid this degradation and ensure high-quality sensor data, denoising methods must be applied.

Image denoising methods, encompassing both color and depth images, are commonly categorized into two primary groups: model-based and learning-based approaches. While learning-based methods can be very powerful, they often require a substantial amount of training data and might carry inherent biases from the training set [Meh+21]. To circumvent these issues, various methods, such as the *joint depth refinement and normal estimation* (JDN) method by Rossi et al. [Ros+20], have emphasized the use of model-based techniques. This chapter aligns with this trend, focusing primarily on image denoising via model-based approaches.

Among the most famous model-based approaches is the total variation (TV) regularization, also known as the Rudin-Osher-Fatemi (ROF) model [ROF92], which utilizes the L_1 -norm of the image gradient as a penalty. The ROF model has been widely used in image denoising and deblurring applications, due to its simplicity, efficiency, and effectiveness in removing noise and blur from images. However, despite its popularity, the ROF model has several significant limitations that restrict its applicability in various imaging scenarios.

One of the major drawbacks of the ROF model is its tendency to favor piecewise constant reconstructions, resulting in what is known as the *staircasing effect* in regions with smooth transitions. This is mainly attributed to the fact that the utilized L_1 regularization is a relaxed variant of the L_0 norm, which in turn favors sparse, i.e. piecewise constant, results.

To address these limitations, non-convex gradient-based regularizers have also been studied and can be motivated by the distribution of natural gradients [HM99]. Notable examples include the famous Mumford-Shah (MS) model [MS89], which is based on a truncated quadratic regularization, and the TV- q penalty [HW13], which is based on an L_q -quasi-norm with $q < 1$.

Simple alternatives to the aforementioned energy minimization methods are plain bilateral filters [TM98], which are a.o. commonly used for denoising depth and normal maps (independently) in 3D reconstruction [Zol+18]. However, these methods also suffer from the limitation that they favor piecewise constant reconstructions and inherently show a staircasing effect in regions of smooth transitions. To address this issue, researchers have attempted to integrate higher-order derivatives, such as through infimal convolutions [CL97], total generalized variation [BKP10; Fer+13], or by estimating affine-linear functions [KSW20]. These techniques partially mitigate the staircase issue and offer promising advancements in enhancing color and depth images.

Exploiting the idea that natural images exhibit *self-similarity* has led to the development of so-called *non-local* denoising techniques. These methods compare entire patches, instead of single pixels, of images. Examples of such algorithms include non-local means [BCM05], block-matching 3D (BM3D) [Dab+07], or patch-based dictionary learning [AEB06]. The *joint depth refinement and normal estimation* (JDN) algorithm by Rossi et al. [Ros+20] falls in this category. It denoises depth images and normal maps jointly, by using 9x9 patches of pixels to exploit non-local information to improve the denoising accuracy.

While non-local methods typically yield higher accuracy in denoising, they come with higher computational demands compared to their local counterparts. This increased computational load arises from the necessity to process relatively large patches per pixel, such as 9x9 patches, as opposed to considering only direct neighbors, which typically involves 3x3 patches. As a result of these higher computational costs, the non-local approaches demonstrate enhanced denoising capabilities, making them valuable for scenarios where accuracy is more important than computational efficiency.

Some approaches operate at an intermediate scale between the local and non-local regime and are based on robustly fitting geometric models, such as planes (e.g., [FTK14; RMB21]) and/or cylinders (e.g., [PG18]). These approaches commonly merge regions of similar models in a region-growing fashion, gradually building larger regions by combining neighboring elements with similar characteristics. However, these region-growing procedures tend to exhibit a greedy behavior, i.e. they may encounter unfavorable partitions during the merging process, leading to less accurate model-fitting results.

It is worth noting that model-based methods inherently exhibit geometric sparsity, which naturally enables significant reductions in data size. As a direct consequence of this property, various applications can leverage the reduced computational demands offered by sparse data, as demonstrated by Geiping et al. [Gei+20]. Their findings indicate that, under specific conditions, computations on sparse data yield results equivalent to those on dense data but with reduced computational costs. This, in turn, facilitates the adoption of more sophisticated yet computationally expensive applications and techniques, such as lifting [Poc+10].

In this chapter, a model-based method is presented that estimates second-order polynomials on the inverse depth map. It will be demonstrated that these models correspond to quadric surfaces and are thus more suitable for rather complex geometries, compared to other model based methods that use simple models such as planes or cylinders. The approach effectively subdivides regions containing different model parameters, while subsequently merging regions with similar models. By adopting this non-greedy approach, the algorithm avoids unfavorable partitions, leading to improved results.

5.3 PROPOSED METHOD

The input to the proposed approach is a RGB-D image consisting of color (or grayscale) image $\mathcal{I}$ and a corresponding depth map $\mathcal{D}$. The color image and depth map are assumed to be temporally synchronized as well as spatially registered, i.e. the color $\mathcal{I}([p_x, p_y])$ corresponds to the depth value $\mathcal{D}([p_x, p_y])$ of the captured scene, for all pixels $[p_x, p_y] \in \Omega$ of the pixel set $\Omega \subset \mathbb{N}^2$.

Subsequently, let $C \in \mathbb{R}^{3 \times 3}$ be the camera matrix of the depth sensor, as defined in equation (2.7)

$$C := \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (5.1)$$

and $\pi(\cdot) : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ be the camera projection, as defined in equation (2.8). Given a depth map $\mathcal{D}(\cdot)$, the corresponding point map $\mathcal{P} : \Omega \rightarrow \mathbb{R}^3$ is computed as in equation (2.13):

$$\mathcal{P}([p_x, p_y]) := C^{-1} \cdot \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \cdot \mathcal{D}([p_x, p_y]) \quad (5.2)$$

Furthermore let

$$\mathcal{N} : \Omega \rightarrow \mathbb{S}^2 := \{\mathbf{n} \in \mathbb{R}^3 : \|\mathbf{n}\|_2 = 1\} \quad (5.3)$$

be a normal map, computed as described in section 2.1.3, and

$$\mathcal{M} : \Omega \rightarrow \mathbb{R}^6 \quad (5.4)$$

be a model map, as described in section 5.3.2.

The procedure of the presented method is sketched in Fig. 5.2, and works as follows: It takes as input one color image $\mathcal{I}$, the corresponding depth image $\mathcal{D}$, and the respective intrinsic camera matrix C of the depth sensor. The depth image is then embedded into a weighted finite graph with a simple 4-neighborhood system, as described in section 2.3. This graph is then iteratively partitioned into disjoint sets of connected components via graph cuts (cf. section 5.3.3), and after each iteration a second-order geometry model (cf. section 5.3.2) is estimated for each partition. The proposed approach uses backward steps to merge similar or small partitions (cf. section 5.3.3) in order to enforce a non-greedy optimization. All these steps are repeated until a stopping criterion is met. The output is a piecewise constant model map $\widetilde{\mathcal{M}}$, which is used to compute a corresponding depth map $\widetilde{\mathcal{D}}$ and its coherent normal map $\widetilde{\mathcal{N}}$ (cf. section 5.3.3).

5.3.1 Inverse Depth Map

Let $\mathcal{P}([p_x, p_y])$ be co-planar points with (unit-length) plane normal $\vec{\mathbf{n}}$. Then through the plane equation from linear algebra and equation (2.13), one can derive

$$d = \vec{\mathbf{n}} \cdot \mathcal{P}([p_x, p_y]) \quad (5.5)$$

$$\Leftrightarrow d = \vec{\mathbf{n}}^\top C^{-1} \cdot [p_x, p_y, 1]^\top \cdot \mathcal{D}([p_x, p_y]) . \quad (5.6)$$

Bringing the depth map $\mathcal{D}([p_x, p_y])$ to the left-hand side leads to

$$\frac{1}{\mathcal{D}([p_x, p_y])} = \frac{1}{d} \vec{\mathbf{n}}^\top C^{-1} \cdot [p_x, p_y, 1]^\top \quad (5.7)$$

$$\Leftrightarrow \frac{1}{\mathcal{D}([p_x, p_y])} = ap_x + bp_y + c , \quad (5.8)$$

with $\frac{1}{d} \vec{\mathbf{n}}^\top C^{-1} = [a, b, c]$ and $a, b, c \in \mathbb{R}$. This means that planar points in 3D (equation (5.5)) correspond to linear functions in the inverse depth image (equation (5.8)), and vice versa. This in turn means that estimations of piecewise linear functions on the inverse depth map will result in piecewise planar functions in the point map. This observation has been made and utilized by various authors, a.o. by Rossi et al. [Ros+20].

5.3.2 Second-order Geometry Model

As discussed in section 5.1 and shown in Fig. 5.1(c), piecewise planar estimations are not sufficient for rather complex geometries. In contrast to prior research the proposal of this chapter is to approximate the inverse depth through a polynomial of second degree:

$$\frac{1}{\mathcal{D}([p_x, p_y])} = ap_x^2 + bp_y^2 + cp_x p_y + dp_x + ep_y + f \quad (5.9)$$

$$\Leftrightarrow \frac{1}{\mathcal{D}([p_x, p_y])} = \mathbf{p} \cdot \mathbf{m} , \quad (5.10)$$

with $\mathbf{p} := [p_x^2, p_y^2, p_x p_y, p_x, p_y, 1]^\top$ and $\mathbf{m} := [a, b, c, d, e, f]^\top$. Subsequently the model parameters $\mathbf{m} \in \mathbb{R}^6$ will be referred to as *second-order model*.

By using equation (2.13) along with $[x, y, z]^\top = \mathcal{P}([p_x, p_y])$, one can derive the following level set equation:

$$0 = \bar{a}x^2 + \bar{b}y^2 + \bar{c}z^2 + \bar{d}xy + \bar{e}xz + \bar{f}yz - z, \quad (5.11)$$

where $\bar{a}, \dots, \bar{f} \in \mathbb{R}$. This means that the proposed *second-order model* corresponds to a (degenerate) quadric with 6 degrees of freedom (DoF) [HZ00]. Compared to previous

plane-based methods that only have 3 DoF, the 6 DoF in the proposed approach makes it better suited for representing geometries with higher complexities.

From the second-order model $\mathbf{m} \in \mathbb{R}^6$, the corresponding depth value can be calculated as the reciprocal of equation (5.10). The coherent (non-unit length) surface normal of the respective geometry can be computed as the derivative of the level set function shown in equation (5.11), i.e. $\mathbf{n} = [\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}]$. This leads to

$$\mathbf{n} = \mathbf{m}^\top \begin{bmatrix} 2f_x p_x & 0 & 2c_x p_x - p_x^2 \\ 0 & 2f_y p_y & 2c_y p_y - p_y^2 \\ f_x p_y & f_y p_x & c_x p_x + c_y p_y - p_x p_y \\ f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (5.12)$$

Normalization of the result yields the corresponding unit-length surface normal $\vec{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|_2}$ of the degenerate quadric represented by the second-order model parameters $\mathbf{m} \in \mathbb{R}^6$.

It is worth to note that the value of surface normals depends on pixel coordinates $[p_x, p_y]$, such that this formula allows to compute a coherent depth map $\tilde{\mathcal{D}}([p_x, p_y])$ and normal map $\tilde{\mathcal{N}}([p_x, p_y])$.

5.3.3 Coherent Depth Image and Normal Map

In order to compute a coherent depth map $\tilde{\mathcal{D}}([p_x, p_y])$ and normal map $\tilde{\mathcal{N}}([p_x, p_y])$, as described in section 5.3.2, a piecewise constant second-order model map $\mathcal{M}([p_x, p_y])$ must be estimated on the input depth map $\mathcal{D}([p_x, p_y])$. This is accomplished by defining the RGB-D input image on a finite weighted graph, as described in section 2.3, and performing a partitioning on this finite weighted graph, similar to the algorithm discussed in section 3.2. The details of this partitioning are described next.

Partitioning on Weighted Finite Graphs

Let $G = (V, E, w)$ be a finite weighted graph, and $\mathbb{P} = \{V_1, \dots, V_N\}$ be the set of disjoint partitions V_i of the vertex set $V = \bigcup_{i=1}^N V_i$, cf. section 2.3. Let $\mathcal{M}_i(v)$ be model maps, which are constant per partition V_i , and zero otherwise:

$$\mathcal{M}_i(v) := \begin{cases} \mathbf{m}_i \in \mathbb{R}^6 & \text{if } v \in V_i \\ \mathbf{0} \in \mathbb{R}^6 & \text{otherwise,} \end{cases} \quad (5.13)$$

where $\mathbf{m}_i$ is the second-order geometric model parameters of partition V_i , as in equation (5.10).

The overall piecewise constant model map $\mathcal{M}_{\mathbb{P}}$ is defined as

$$\mathcal{M}_{\mathbb{P}}(v) := \sum_{i=1}^N \mathcal{M}_i(v), \quad (5.14)$$

which is a piecewise constant function of second-order model parameters over the partition set $\mathbb{P}$.

The objective of the proposed optimization approach is twofold. Firstly, it aims to partition regions based on their model parameters. Secondly, it seeks to (robustly) estimate model parameters for these partitions. This *chicken-and-egg problem* can be formulated as:

$$\min_{\mathcal{M}_{\mathbb{P}}} \sum_{v \in V} \phi_D(\mathcal{M}_{\mathbb{P}}(v)) + \alpha \sum_{[v,u] \in E} \phi_0((\nabla_w \mathcal{M}_{\mathbb{P}})(v, u)). \quad (5.15)$$

The data term ϕ_D is defined as

$$\phi_D(\mathcal{M}(v)) = \phi_{L_2^2} \left(\mathbf{p} \cdot \mathcal{M}(v) - \frac{1}{\mathcal{D}(v)} \right), \quad (5.16)$$

where pixel $[p_x, p_y] \in \Omega$ and vertex $v \in V$ correspond to each other, the function $\phi_{L_2^2}(\mathbf{x}) := \frac{1}{2} \|\mathbf{x}\|_2^2$ is the squared L_2 norm, and $\mathbf{p} := [p_x^2, p_y^2, p_x p_y, p_x, p_y, 1]^\top$.

The regularization term is based on the L_0 “norm”, introduced in equation (2.59)

$$\phi_0(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} = \mathbf{0} \\ 1 & \text{otherwise} \end{cases}. \quad (5.17)$$

This forces neighboring pixels/vertices to have (piecewise) constant model parameter values.

Given the data and regularization terms, it can be shown that equation (5.15) is a so-called minimal partitioning problem on finite graphs [TGB19], which can be solved via Cut Pursuit algorithm proposed by Landrieu and Obozinski [LO17], similar to section 3.2. The standard Cut Pursuit approach is used in a modified way, which is described next.

Modified Cut Pursuit

The solution of the underlying problem equation (5.15) is iteratively computed as follows. Let $k \in \mathbb{N}_0$ be the index of one particular iteration, $G_k = (V, E_k, w)$ be a finite graph at iteration k , initialized with $G_0 = G$, and $\mathbb{P}_k = \{V_1, \dots, V_{|\mathbb{P}_k|}\}$ be a partitioning of V at iteration k , initialized with $\mathbb{P}_0 = \{V\}$, i.e. initially all vertices of the graph lie in the same partition. Let $\mathcal{M}_{\mathbb{P}_k}$ be the piecewise constant model map at iteration k , as defined in equation (5.14), initialized as the solution of the linear equation arising from $\mathcal{M}_{\mathbb{P}_0} = \arg \min_{\mathcal{M}_{\mathbb{P}_0}} \sum_{v \in V} \phi_{L_2^2}(\mathcal{M}_{\mathbb{P}_0}(v))$.

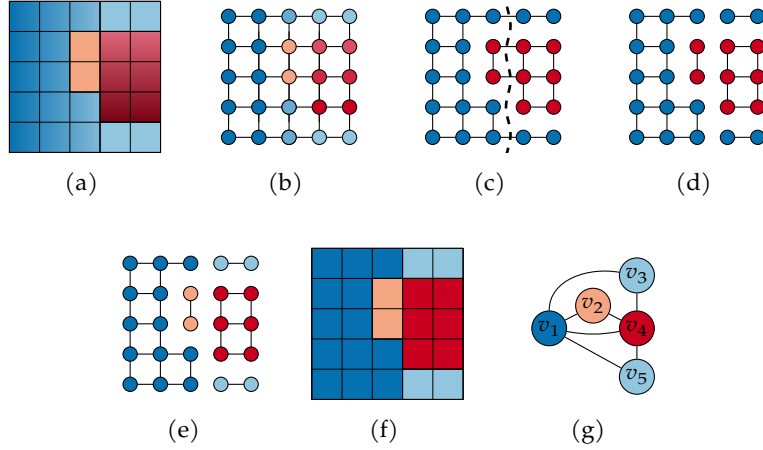


Figure 5.2: Workflow of the Cut Pursuit algorithm, where model parameters are represented in colors. The input data (a) is transformed via a 4-neighborhood to a finite graph (b). Graph cuts are applied in order to partition similar values (c) and (d). For each partition a new model is estimated (d). This process is repeated until no further partitioning can be done. The result contains piecewise-constant values of model parameters (f) and can be represented as a finite graph (g).

Iteratively applying the following steps will minimize equation (5.15):

1. Binary partitioning of $d(v) = \langle \nabla_w \phi_D(\mathcal{M}_{\mathbb{P}_k}(v)), \gamma \rangle$ with $\gamma = \mathbf{1} \in \mathbb{R}^n$:

$$\begin{aligned} \min_{\rho} \sum_{v \in V} d(v) \cdot \rho(v) + \alpha \sum_{[u,v] \in E_k} |(\nabla \rho)(u,v)| \\ \text{s.t. } \rho(v) \in [0, 1], \end{aligned} \quad (5.18)$$

which can be solved as discussed in section 3.2.

2. Set $B = \bigcup_{v \in V} v$ for $\rho(v) \geq 0.5$ and conversely $\bar{B} = \bigcup_{v \in V} v$ for $\rho(v) < 0.5$, which, by the thresholding theorem [CEN06], minimizes equation (5.18) over binary ρ .
3. Remove edges connecting B and $\bar{B} = V \setminus B$ from edge set E_k :

$$E_{k+1} = E_k \setminus \{[u,v] : u \in B, v \in \bar{B}\} \quad (5.19)$$

4. Given E_{k+1} , compute new connected component partitions of V , which results in a new partition set $\mathbb{P}_{k+1}$.
5. Estimate models $\mathbf{m}_i \in \mathbb{R}^6$ on partitions $V_i \in \mathbb{P}_{k+1}$ by solving

$$\mathcal{M}_{\mathbb{P}_{k+1}} = \arg \min_{\widetilde{\mathcal{M}}_{\mathbb{P}_{k+1}}} \sum_{v \in V} \phi_H(\widetilde{\mathcal{M}}_{\mathbb{P}_{k+1}}(v)) \quad (5.20)$$

where ϕ_H is the Huber norm [Hub64].

These steps are illustrated in Fig. 5.2. This procedure is repeated until no further partitioning can be performed in step (1), i.e. either $B = \emptyset$ or $\bar{B} = \emptyset$, or a maximum number of iterations $k > t_{iter}$ has been reached.



Figure 5.3: Illustration of a merge step. The two blue partitions in (a) have the same value, i.e. they can be merged. During merging, connecting edges between the two partitions are added to the edge set E_k . After this a new model parameter value is estimated for the entire (merged) partition.

Note that the proposed version of the Cut Pursuit algorithm operates on different norms, namely on $\phi_{L_2^2}$ and ϕ_H , in step (1) and step (5). This allows for simple computation of the directional derivative as well as robust model fit in steps (1) and (5), respectively. This is a key difference to the original Cut Pursuit algorithm by Landrieu and Obozinski [LO17], where both norms must be equal. Furthermore the proposed formulation differs to the work of Tenbrinck et al. [TGB19], who motivated the use of two different regularization terms.

Merge

To guarantee the existence of unique solutions in step 5 from the preceding section and to minimize uncertainties in model fitting, small partitions are merged with their best-matching adjacent larger partitions. Therefore, at the end of each iteration step $k \in \mathbb{N}_0$, the proposed method iterates over all partitions V_i and compares the sizes with the adjacent partitions V_j . If $|V_i| < t_{minSize}$ whilst $|V_j| \geq t_{minSize}$, with parameter $t_{minSize} \geq 6$, then these two partitions are marked as *merge pair candidates*. Subsequently the pair $[V_i, V_j]$ is merged with the least error $\sum_{v \in V_i} \phi_H(\tilde{\mathcal{D}}(v) - \mathcal{D}(v))$, with the current depth estimation $\tilde{\mathcal{D}}$ given model $\mathbf{m}_j$ and the input depth image $\mathcal{D}$. All merge pair candidates which include V_i or V_j are then updated and again the pair with the least error is merged. This procedure, illustrated in Fig. 5.2, is repeated until all merge pairs are processed.

Likewise partitions with similar models are merged, where a merge pair candidate is selected if $\|\mathbf{m}_i - \mathbf{m}_j\| < t_{merge}$. The merge pair candidates with the smallest error $\|\mathbf{m}_i - \mathbf{m}_j\|$ are merged. Note that this simplistic error metric does not consider the diverse ranges of the elements in $\mathbf{m}_i \in \mathbb{R}^6$. Therefore, adopting a more sophisticated metric, such as incorporating the Mahalanobis distance [Mah36] or comparing partition elements, as proposed by Garland and Heckbert [GH97], will likely yield better results. However, incorporating these methods would involve additional iterations through the partitions elements, resulting in more frequent memory accesses, leading to a worse runtime performance. Thus the decision towards the naive metric was driven by runtime performance considerations.

5.4 EVALUATION

In this section the performance of the proposed approach is discussed. The focus of this evaluation will be on the quality of depth images and normal maps. This is in contrast to other research, e.g. Kornilova et al. [Kor+22], which analyzed the accuracy of the resulting partitions instead, and ignored the quality of the resulting values.

To measure the quality of the results, the mean, median, and root mean squared error (RMSE) between the estimated depth values and the ground truth depth images serve as one measure. The second measure is based on the mean, median, and RMSE of the angle error between the estimated normals and the ground truth normals. Only valid estimates will be considered, i.e. if any values are missing in the results, they are excluded from the evaluation. Thus the third and last measure of quality is the number of valid estimates in the results.

The proposed method is compared to the following approaches

- *Cylinder and Plane Extraction (CAPE)* by Proença and Gao [PG18] uses a region growing approach on the depth values to estimate plane and cylinder parameters and to partition RGB-D images according to the estimated parameters.
- *Depth Dependent Planar Flood Fill (DDPFF)* by Roychoudhury et al. [RMB21] uses plane estimations on the depth map along with a flood fill to partition the RGB-D images.
- *Joint Depth Refinement and Normal Estimation (JDN)* by Rossi et al. [Ros+20] uses a non-local approach (9x9 patches) to estimate plane parameters on the inverse depth map and then return the normals from the plane estimation.
- *Parallel Algorithm for the Piecewise Affine-Linear Mumford-Shah Model (PALMS)* by Kiefer et al. [KSW20] estimates linear functions on color images and clusters those.

As discussed in section 5.3.2, piecewise planar functions in 3D can be approximated as piecewise linear functions of inverse depth. Therefore the evaluation in this section also includes **PALMS** algorithm, which will be performed to compute piecewise linear functions on inverse depth images, i.e. piecewise planar models on depth images.

The presented analysis relies on datasets with ground truth normals. Therefore the synthetic dataset ICL NUIM [Han+14] as well as the real-world dataset NYUv2 [Sil+12] are utilized. The ICL NUIM dataset provides synthetic depth images without noise, which are used as ground truth. Additionally this dataset contains depth images with simulated Kinect sensor-noise. The NYUv2 dataset contains challenging real-world scenes captured with a Kinect camera. It provides estimated normal images [LZP14], which are used to evaluate the accuracy of normals. In all experiments the parameters are set as follows $\alpha = 2.5$, $t_{iter} = 6$, and $t_{minSize} = 16$.

The experiments are performed on a machine equipped with 16GB RAM, an Intel Core i7-11800H CPU, and a NVIDIA GeForce RTX 3070 GPU.

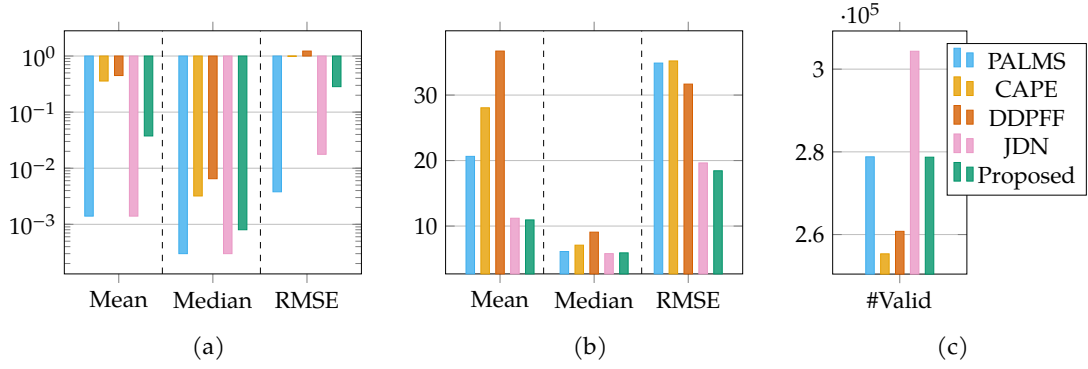


Figure 5.4: Results on clean ICL NUIM dataset, with (a) the depth error (lower is better), (b) normal angle errors (lower is better), and (c) the number of estimated valid points (higher is better).

5.4.1 Evaluation on Clean ICL NUIM Dataset

The ICL NUIM dataset [Han+14] contains depth images without noise and missing values. These clean depth images serve also to compute normal maps via finite forward differences, as described in section 2.1.3.

The algorithms **PALMS**, **CAPE**, **DDPFF**, **JDN**, and the proposed approach are performed on this dataset. The results are shown in Fig. 5.4. The runtimes for each algorithm are presented in table 5.1.

Algorithm	Programming Language	Runtime [s]
CAPE [PG18]	C++	0.1
DDPFF [RMB21]	C++	0.3
JDN [Ros+20]	CUDA and Python wrapper	14
PALMS [KSW20]	C++ and Matlab wrapper	31
Proposed	Matlab	3

Table 5.1: Programming languages and runtimes.

Depth errors between the estimation and the ground truth, i.e. the original clean depth values, are shown in Fig. 5.4(a). As can be seen, the algorithms **PALMS**, **JDN** and the proposed approach outperform **CAPE** and **DDPFF**.

Normal angle errors between the estimation and the computed normals from finite differences of the ground truth depth, are shown in Fig. 5.4(b). These values are best approximated by **JDN** and the proposed approach.

As discussed, the mentioned metrics do not consider invalid estimations, i.e. such which contain *NaN* (not a number) or depth values which are non-positive. The number of valid pixels per algorithm is shown in Fig. 5.4(c).

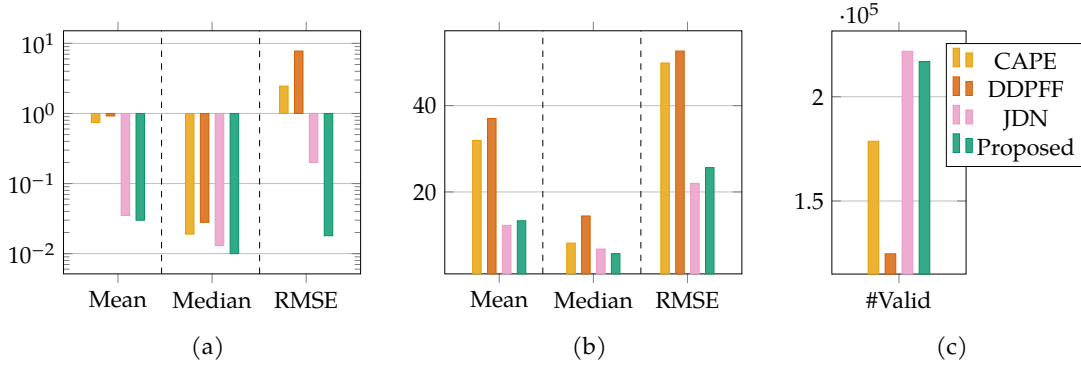


Figure 5.5: Results on noisy ICL NUIM dataset, with (a) the depth error, (b) normal angle errors, and (c) the number of estimated valid points.

Considering quality metrics only, then algorithm **JDN** [Ros+20] outperforms all others. However, it is essential to highlight that despite the parallel processing advantage of running on a GPU (CUDA), **JDN** exhibits a comparably slow runtime performance. This is due to the algorithm’s non-local nature of using 9×9 patches, coupled with a substantial number of iterations. Furthermore it is also worth to note the on-par performance of **PALMS** [KSW20], which was actually designed to estimate linear functions on color images instead of (inverse) depth maps. Taking the runtime from table 5.1 also into consideration, then the proposed approach provides a good trade-off between accuracy and runtime performance.

5.4.2 Evaluation on Noisy ICL NUIM Dataset

The ICL NUIM dataset [Han+14] contains also depth images with simulated Kinect noise, which are used for evaluation in this section. This allows for analysis on more realistic data, paired with ground truth information from clean synthetic data. The clean depth maps will serve as ground truth for the evaluation of depth related metrics. Additionally, similar to the previous section, normal maps are computed via finite forward differences from clean depth maps, and serve as ground truth for the evaluation of normals related metrics.

As before, the algorithms **CAPE**, **DDPFF**, **JDN**, and the proposed approach are performed on this dataset. The results are shown in Fig. 5.5. The runtimes did not change and equal those shown in table 5.1. It is worth to note that in this scenario **PALMS** failed to compute valid estimations on almost all depth images, which is why it was excluded from the results. As mentioned, this algorithm was tailored for color images, which generally exhibit a lower level of disturbance in comparison to depth images. Consequently, the simulated Kinect noise, characterized by extreme outliers and missing values, caused this algorithm to fail.

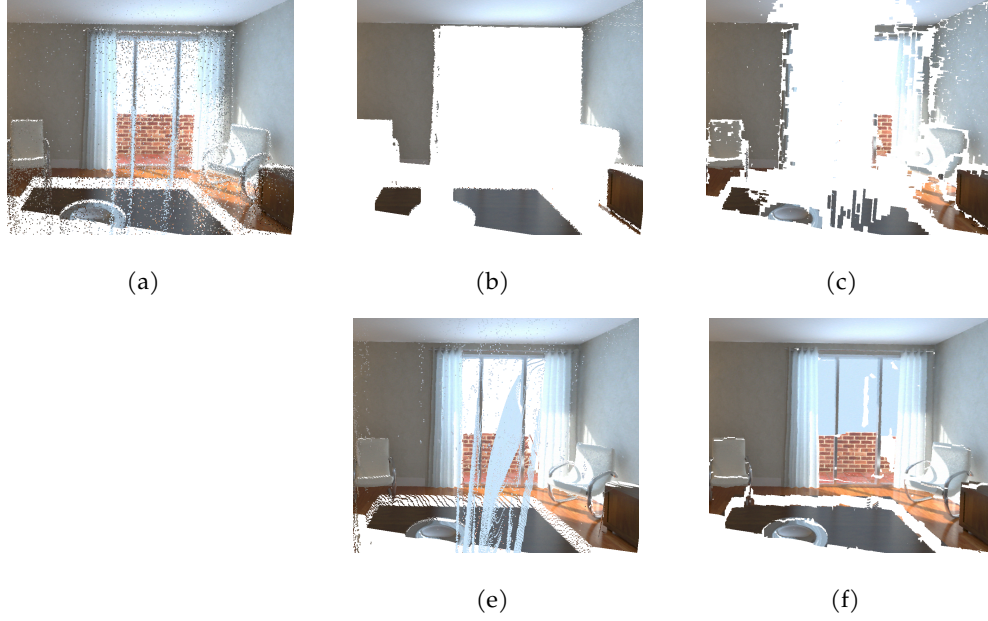


Figure 5.6: Result (shown as point cloud) of particular frame from the noisy ICL NUIM dataset. The noisy input is shown in (a), together with results of **CAPE** (b), **DDPFF** (c), **JDN** (d) and the proposed approach (e). As can be seen, the proposed method yields highly dense and accurate result, with fewer outliers compared to the other algorithms.

Depth errors between the estimation from noisy input and the ground truth, i.e. the original clean depth values, are shown in Fig. 5.5(a). As can be seen, **JDN** and the proposed approach outperform **CAPE** and **DDPFF**.

Normal angle errors between the estimation and the computed normals from finite differences of the ground truth depth, are shown in Fig. 5.5(b). These values are best approximated by **JDN** and the proposed approach.

As discussed, the mentioned metrics do not consider invalid estimations, i.e. such which contain *NaN* (not a number) or depth values which are non-positive. The number of valid pixels per algorithm is shown in Fig. 5.5(c).

Considering quality metrics only, then **JDN** [Ros+20] and the proposed approach provide similar results. However, the proposed algorithm has a significantly smaller runtime compared to **JDN**, as discussed in section 5.4.1.

A quantitative comparison is shown in Fig. 5.6. It can be observed that **JDN** and the proposed approach provide dense results, i.e. depth (and normal) maps with least missing values, which reflects the quantitative data in Fig. 5.5(c). A closer look at Fig. 5.6(e) reveals that the **JDN** result exhibits more outliers, compared to the proposed approach. This is also present in the quantitative data in Fig. 5.5(a), as the RMSE for the proposed approach is significantly lower.

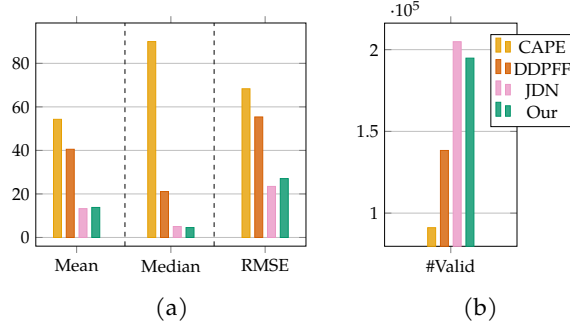


Figure 5.7: Results on NYUv2 dataset, with (a) normal angle errors, and (b) the number of estimated valid points. Note that **CAPE** failed on most frames, which is why its median error is at 90° .

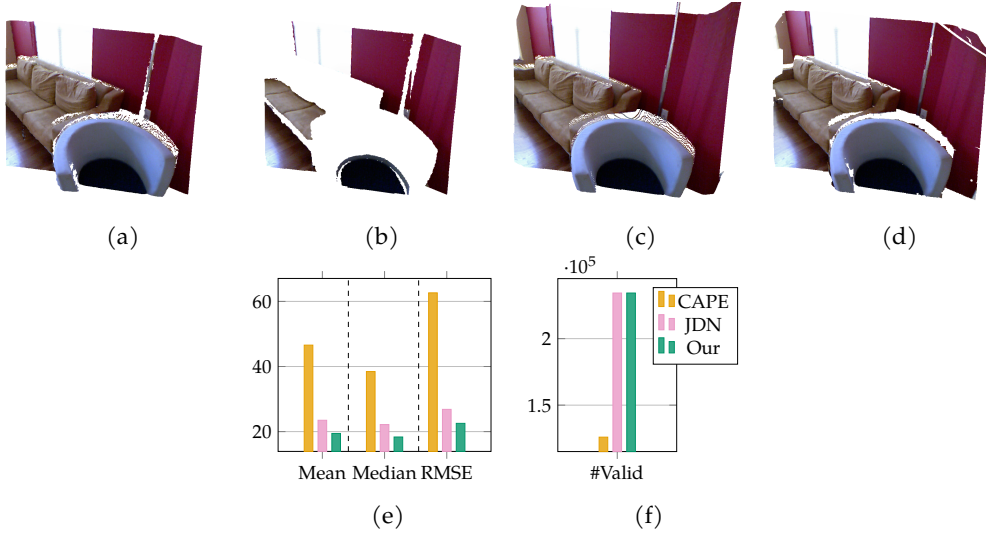


Figure 5.8: Result (shown as point cloud) of one particular frame from the NYUv2 dataset. The noisy input is shown in (a), together with results of **CAPE** (b), **JDN** (c) and the proposed approach (d). **DDPFF** was excluded, as it failed on this input. Instead the normal errors for this frame (e) and number of valid points (f) are shown.

5.4.3 Evaluation on Real-World NYUv2 Dataset

The quantitative results on the real-world NYUv2 dataset, which contains complex and therefore challenging scenes, are shown Fig. 5.7. As this dataset does not provide ground truth of depth, only the errors regarding normals are shown. However, since these normals are estimated via [LZP14] and thus do not represent absolute ground truth, the resulting quantitative metrics should be treated with caution. This dataset has the same resolution as the ICL NUIM dataset and thus the runtime performance is similar to the one shown in table 5.1. One qualitative example is given in Fig. 5.8. Similar to the previous experiments on ICL NUIM, **JDN** and the proposed approach provide the best results.

5.5 CONCLUSION

In this chapter, a new method for coherent denoising of depth maps and normal maps was proposed. This method is based on iteratively finding partitions in such a way that the depth map on each partition can be well-represented by a (degenerate) quadric, which is obtained by fitting second-order model parameters to the inverse depth. As a consequence, the proposed approach can represent the entire geometry of a scene by a sparse set of coefficients. These values are then used to produce dense and denoised maps of coherent depth and normal values. Numerical experiments demonstrate that the proposed technique outperforms existing methods in terms of accuracy and density, particularly in scenes with complex geometry. It provides dense and accurate results, similar to JDN [Ros+20], but with better runtime performance. Furthermore it partitions the data, similar to CAPE [PG18], DDPFF [RMB21], and PALMS [KSW20]. These clusters can be used for data reduction and thus it permits the use of advanced yet computational expensive techniques, like lifting.

FL0C: FAST L_0 CUT PURSUIT

In this chapter, a stochastic algorithm is introduced to efficiently approximate graph cuts in the Cut Pursuit algorithm. The proposed approach exhibits an improved convergence rate, thorough a combination of a novel convex formulation of the underlying graph cut problem and a stochastic solver. This enhancement results in faster runtime performance for Cut Pursuit-based algorithms, as introduced in section 3.2 and applied in chapter 5.

This chapter begins with a brief introduction of image partitioning, explaining the mathematical challenges and the need for improved runtime performance. It then covers the derivation of the proposed stochastic algorithm, followed by an evaluation of its convergence and runtime efficiency.

The approach discussed in this chapter was published [GMK22] and presented at the IEEE International Conference on Image Processing (ICIP) 2022 in Bordeaux, France.

6.1 MOTIVATION

Image partitioning is a fundamental task in image processing and computer vision, which focuses on segmenting an image into distinct regions, each characterized by similar attributes. These attributes can be homogeneous colors, as highlighted in section 3.2, or more complex features, such as the identical model parameters, as discussed in chapter 5. The inherent sparsity of minimal partitioning results can be leveraged to accelerate computationally demanding algorithms, as demonstrated by Geiping et al. [Gei+20].

The algorithms previously discussed in this thesis exhibit certain limitations. The algorithm presented in chapter 4, which utilizes the *truncated quadratic “norm”*, achieves real-time performance but fails to provide piecewise constant results without additional post-processing. Conversely, the algorithm proposed in chapter 5 successfully delivers piecewise constant results but at the cost of computational efficiency.

This chapter discusses an acceleration of the Cut Pursuit algorithm. The focus is on enhancing the speed of the graph cut algorithm involved in each Cut Pursuit iteration. The goal is to maintain the desirable piecewise constant results while significantly improving computational performance, combining the strengths of the previously discussed methods.

6.2 RELATED WORK

Despite being one of the first analyzed image processing problems, image partitioning is still an active field of research, mainly due to its $\mathcal{NP}$ -Hard optimization complexity and its close relation to other computer vision tasks [Poc+08; GSC13; Ye+21]. Consequently there is a variety of algorithms [NTC13; Kap+15; BK04] that aim to solve this problem. Those approaches mainly differ in the definition of ϕ_D , the computation of the perimeter and/or the applied solver.

One of the most prominent examples is the piecewise constant Mumford-Shah problem [MS89], where the term $\phi_D(x)$ is defined as follows

$$\phi_D(x) = \|c_i - f(x)\|_2^2, \quad (6.1)$$

with $c_i = \frac{1}{|P_i|} \sum_{x \in P_i} f(x)$ being the mean value of partition P_i . This formulation plays a central role in numerous image partitioning methods and has been widely adopted in various applications, such as the estimation of piecewise constant colors, discussed in chapter 3. A simpler alternative was introduced by Rudin, Osher, and Fatemi (ROF) [ROF92], which replaces the perimeter term with the L_1 norm, providing a more relaxed and simplified version of the optimization problem.

Geiping et. al. [Gei+20] analyzed the relationship between optimization problems on full dense images (cf. Fig. 6.1(a)) and their corresponding reduced graph representations (cf. Fig. 6.1(b)). They prove that if the jump set of the dense image is a subset of the jump set of the graph representation, then both optimization problems lead to same results. The impact of this finding is twofold, firstly the reduced setting allows to compute the same result more efficiently, and secondly it permits the use of advanced yet computational expensive techniques, such as lifting [Poc+10]. Lang et al. [LSV21] showed that image partitioning can be solved with Graph Cut based algorithms near optimally, despite being a $\mathcal{NP}$ -Hard problem. This, in turn, means that Graph Cut partitions and more importantly their borders will (mostly) coincide with input data and energy optimization solutions, and thus obey the requirements mentioned by Geiping et al. [Gei+20].

However, despite these advancements, most modern Graph Cut based multi-partitioning approaches, including α -expansions [BVZ01] and Cut Pursuit [LO17], still involve comparably high computational costs. These costs can negate some of the efficiency gains achieved through the use of sparse representations.

This chapter discusses a fast multi-partitioning approach based on Cut Pursuit [LO17], in a variational variant, similar to [TGB19]. This approach improves the runtime performance by convexifying [BER17] the underlying variational graph cut problem, coupled with the usage of a stochastic primal dual hybrid gradient (SPDHG) solver [Cha+18]. The results reveal a significant improvement of runtime performance at the same accuracy as comparable methods. It is worth to note that this approach is per se parallelizable without additional modifications [RL19].

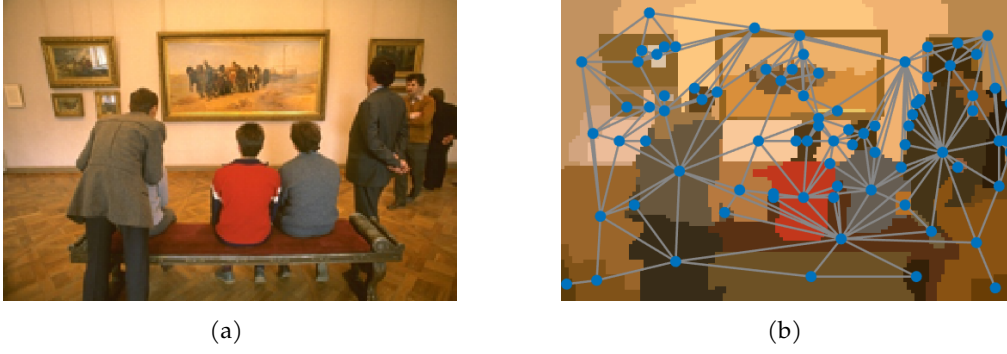


Figure 6.1: The original image (a) and its piecewise constant approximation with the corresponding graph (b). Note that the graph consists of only 82 vertices, which is a significant reduction compared to the original 481×321 pixels.

6.3 PROPOSED METHOD

The input to the proposed method is a RGB image $\mathcal{I} : \Omega \rightarrow \mathbb{R}_+^3$, where Ω represents the image domain (set of pixels). Notably, while RGB images are chosen for simplicity, the approach is equally applicable to RGB-D images, supporting their integration into RGB-D based image partitioning.

This work builds on concepts from chapter 5 by focusing on the Cut Pursuit method [LO17], which estimates solutions to equation (3.2) for arbitrary differentiable functionals ϕ_D . Starting with one big partition for the entire image, i.e. $P_1 = \Omega$ and $\mathbb{P}_0 = \{P_1\}$, this algorithm iteratively subdivides each partition $P_i \in \mathbb{P}_j$ into smaller disjoint parts, leading to a larger partition set $\mathbb{P}_{j+1}$. This partitioning is performed via graph cuts [BK04; BK03], which solve equation (5.15) for $N = 2$. The entire subdivision process is repeated until no further partitioning can be performed, i.e. $\mathbb{P}_j = \mathbb{P}_{j+1}$. The final result is an (approximate) solution of equation (5.15), for an a priori unknown number of partitions.

The overall procedure is embedded in a finite weighted graph $G = (V, E, w)$, where vertices $v \in V$ represent centroids of partitions P_i , and edges $(u, v) \in E$ connecting vertices $u, v \in V$ whose corresponding partitions P_i, P_j share a border, i.e. $\partial P_i \cap \partial P_j \neq \emptyset$. The resulting graph is a significantly reduced data representation of the given dense input image, as shown in Fig. 6.1.

To accelerate the overall computation, two approaches will be employed: first, the optimization problem will be convexified to improve convergence, similar to the work of Berkels et al. [BER17]. Second, a Stochastic Primal Dual Hybrid Gradient (SPDHG) algorithm, as proposed by Chambolle et al. [Cha+18], will be utilized. These methods will be explained in detail in this chapter.

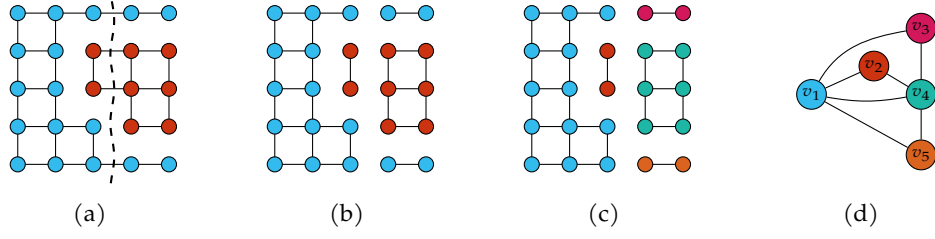


Figure 6.2: Procedure of Cut Pursuit on a given graph with two partitions. The graph is divided via graph cuts (dashed line) (a) and the cut edges are removed (b). The vertices are clustered with connected components and new values are approximated (c). The reduced graph (d) can be deduced from (c).

6.3.1 Fast Cut Pursuit

Landrieu and Obozinski [LO17] proposed the Cut Pursuit algorithm, which finds an estimation of the minimal partition problem equation (3.2), for differentiable functionals ϕ_D , such as the Mumford-Shah functional introduced in equation (6.1). In this chapter a variant of Tenbrinck et al. [TGB19] is utilized instead, which works as follows.

Let $k \in \mathbb{N}_0$ denote the current iteration, $G_k = (V, E_k, w)$ be a finite graph, initialized with $G_0 = G$, and $\mathbb{P}_k = \{P_1, \dots, P_{|\mathbb{P}_k|}\}$ be a partitioning of V , initialized with $\mathbb{P}_0 = \{V\}$, i.e. all vertices belong to one single partition. Let $f_{\mathbb{P}_k}$ be a piecewise constant function, initialized with $f_{\mathbb{P}_0} = \arg \min_{\tilde{f}_{\mathbb{P}_0}} \sum_{v \in V} \phi_D(\tilde{f}_{\mathbb{P}_0}(v), g(v))$. Given the function $g(\cdot)$, Cut Pursuit estimates the minimizer $f_{\mathbb{P}}$ of equation (3.2) through following steps:

- (1) Binary ($N = 2$) partitioning of the directional derivative
 $d(v) = \langle \nabla \phi_D(f_{\mathbb{P}_k}(v), g(v)), \gamma \rangle$ with $\gamma = \mathbf{1} \in \mathbb{R}^n$:

$$\min_p \sum_{v \in V} d(v) \cdot \rho(v) + \lambda \sum_{(u,v) \in E_k} |(\nabla \rho)(u, v)|, \quad (6.2)$$

where $\rho(\cdot)$ is the indicator function of a binary partition $B \subseteq V$, and thus $\|\nabla \rho\|_1 = \text{Per}(B)$ [CCP12].

- (2) Remove edges connecting B and $\bar{B} = V \setminus B$ from EE_k :

$$E_{k+1} = E_k \setminus \{(u, v) : u \in B, v \notin B\} \quad (6.3)$$

- (3) Compute new partitioning $\mathbb{P}_{k+1}$ of V via connected components over E_{k+1} .

- (4) Compute $f_{\mathbb{P}_{k+1}} = \arg \min_{\tilde{f}_{\mathbb{P}_{k+1}}} \sum_{v \in V} \phi_D(\tilde{f}_{\mathbb{P}_{k+1}}(v), g(v))$.

These steps are illustrated in Fig. 6.2. They are repeated until in step (1) no further partitioning can be performed, i.e. either $B = \emptyset$ or $\bar{B} = \emptyset$.

Subsequently equation (6.2) will be replaced as described in the next section, which yields our *Fast L_0 Cut Pursuit*.

6.3.2 Fast Binary Partitioning

The main disadvantage of the standard binary partitioning formulation, given in equation (6.2), is the first term, which negatively influences the convergence properties and thus slows down the entire optimization process. A further drawback lies in the implicit estimation of the indicator function $\bar{\rho}(v) = 1 - \rho(v)$ of partition $\bar{B} = V \setminus B$, which may decrease the accuracy of the estimated results and also slows down the convergence rate [CCP12].

Following the advice of Chambolle et al. [CCP12], in the proposed approach both characteristic functions ρ and $\bar{\rho}$, which are denoted ρ_1 and ρ_2 respectively, will be estimated. Furthermore the left term of equation (6.2) is convexified, similar to the proposal by Berkels et al. [BER17]. Given $N = 2$, and $\omega(v) = |\nabla d(v)|$, the convex binary partitioning problem reads

$$\min_{\rho_i} \sum_{i=1}^N \sum_{v \in V} \omega(v) (\rho_i(v) - \iota_i(v))^2 + \frac{\lambda}{2} \sum_{(u,v) \in E} |(\nabla \rho_i)(u, v)| \quad (6.4)$$

$$\text{s.t. } \sum_{i=1}^N \rho_i(v) = 1 \quad \forall v \in V. \quad (6.5)$$

The initial indicator functions ι_i are defined as

$$\iota_1(v) := \begin{cases} 1 & d(v) < 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.6)$$

$$\iota_2(v) := \begin{cases} 1 & d(v) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.7)$$

$$\Leftrightarrow \iota_2(v) := 1 - \iota_1(v) \quad (6.8)$$

It is worth to note that there is no need to postulate $\rho_i(v) \geq 0$, which would increase the overall complexity of the optimization problem. This is mainly due to the utilization of ι_i functions, which pull the values of ρ_i towards 0 or 1 during optimization.

The optimization problem shown in equations (6.4) and (6.5) can be rewritten by setting $\rho(v) = [\rho_1(v), \rho_2(v)]^\top$ and $\iota(v) = [\iota_1(v), \iota_2(v)]^\top$, which leads to

$$\min_{\rho} \sum_{v \in V} \omega(v) (\rho(v) - \iota(v))^2 + \frac{\lambda}{2} \sum_{(u,v) \in E} \|\nabla \rho(u, v)\|_1 \quad (6.9)$$

$$\text{s.t. } [1, 1] \cdot \rho(v) = 1 \quad \forall v \in V. \quad (6.10)$$

This variational optimization problem can be solved with PDHG approach, as described next.

6.3.3 PDHG Solver

In order to be able to solve equations (6.9) and (6.10) with PDHG algorithm (cf. section 2.4.5), these equations must be rewritten to identify ϕ_D and ϕ_F :

$$\min_{\rho} \sum_{v \in V} \underbrace{\omega(v) (\rho(v) - \iota(v))^2}_{\phi_D} + \underbrace{\frac{\lambda}{2} \sum_{(u,v) \in E} \|\nabla \rho(u, v)\|_1 + \delta_{=1}([1, 1] \cdot \rho(v))}_{\phi_F} . \quad (6.11)$$

Note that term $\phi_F(\cdot)$ can be thought of as a two parameter function of the sum of a L_1 norm and a *equals one* indicator function (cf. section 2.2.4)

$$\phi_F(\mathbf{x}, \mathbf{y}) = \underbrace{\frac{\lambda}{2} \|\mathbf{x}\|}_{\phi_{\frac{\lambda}{2} L_1}(\mathbf{x})} + \delta_{=1}(\mathbf{y}), \quad (6.12)$$

with $\mathbf{x} = \nabla_w \rho(v)$ and $\mathbf{y} = [1, 1] \cdot \rho(v)$, and thus

$$K_v = \begin{bmatrix} \nabla_w \\ [1, 1] \end{bmatrix} . \quad (6.13)$$

From separable sum theorem of convex conjugates (cf. section 2.4.1 and equation (2.74)) follows that the convex conjugate ϕ_F^* is the sum of $\phi_{L_1}^*(\cdot)$ and $\phi_{\delta_{=1}}^*(\cdot)$.

The convex conjugate of the L_1 norm was given in equation (2.80), along with the scaling property in equation (2.75) it is

$$\phi_{\frac{\lambda}{2} L_1}^*(\mathbf{x}) = \frac{\lambda}{2} \left\| \frac{2}{\lambda} \mathbf{x} \right\|_1^* \quad (6.14)$$

$$= \begin{cases} 0 & \left\| \frac{2}{\lambda} \mathbf{x} \right\|_{\infty} \leq 1 \\ \infty & \text{otherwise} \end{cases} \quad (6.15)$$

$$= \delta_{\|\cdot\|_{\infty} \leq \frac{\lambda}{2}}(\mathbf{x}) \quad (6.16)$$

The convex conjugate of $\delta_{=1}(\cdot)$ is derived as follows

$$\delta_{=1}^*(x) = \sup_{y \in \mathbb{R}} yx - \delta_{=1}(y) \quad (6.17)$$

$$= \sup_{y=1} yx \quad (6.18)$$

$$= x \quad (6.19)$$

$$= \text{id}(x) . \quad (6.20)$$

The proximal mapping of $\phi_{\frac{\lambda}{2} L_1}^*(\mathbf{x})$ follows from remark 2.18

$$\text{prox}_{\sigma \delta_{\|\cdot\|_{\infty} \leq \frac{\lambda}{2}}}(\mathbf{y}) = \text{proj}_{\frac{\lambda}{2} \mathbb{B}_{\infty}}(\mathbf{y}) , \quad (6.21)$$

and the proximal mapping of $\delta_{=1}^*(x) = \text{id}(x)$ follows from equations (2.93) and (2.94) with $g(x) = 0$ and $a = \sigma$

$$\text{prox}_{\sigma \text{id}(\cdot)}(y) = y - \sigma. \quad (6.22)$$

Together with the separable sum theorem of the proximal operator (cf. equation (2.89)) follows the proximal mapping of $\phi_F^*(\cdot)$:

$$\text{prox}_{\sigma \phi_F^*}(\mathbf{y}, z) = \begin{bmatrix} \text{prox}_{\sigma \delta_{\|\cdot\|_\infty \leq \frac{\lambda}{2}}}(\mathbf{y}) \\ \text{prox}_{\sigma \text{id}(\cdot)}(z) \end{bmatrix} \quad (6.23)$$

The proximal mapping of ϕ_D is

$$\text{prox}_{\tau \phi_D}(\mathbf{y}) = \frac{\mathbf{y} + 2\tau\omega\mathbf{l}}{1 + 2\tau\omega}. \quad (6.24)$$

Algorithm 6.1, which is equal to PDHG Alg1 (cf. algorithm 2.3), along with equations (6.23) and (6.24) is used to solve equation (6.11).

Algorithm 6.1 PDHG "Alg1" Solver of Equation (6.11)

- 1: Choose $\tau, \sigma > 0$ (with $\tau\sigma\|K\|_{\text{op}}^2 \leq 1$), and relaxation parameter $\theta \in [0, 1]$
 - 2: **repeat**
 - 3: **for each** $v \in V$ **do**: $y^{k+1}(v) = \text{prox}_{\sigma \phi_F^*}(y^k(v) + \sigma K_v \bar{\rho}^k(v))$
 - 4: **for each** $v \in V$ **do**: $\rho^{k+1}(v) = \text{prox}_{\tau \phi_D}(\rho^k(v) - \tau K_v^\top y^{k+1}(v))$
 - 5: **for each** $v \in V$ **do**: $\bar{\rho}^{k+1}(v) = \rho^{k+1}(v) + \theta(\rho^{k+1}(v) - \rho^k(v))$
 - 6: **until** convergence or $k > k_{\max}$
-

6.3.4 Stochastic Solver

In order to speed up the computation of the PDHG Algorithm (cf. algorithm 6.1), the *Stochastic Primal Dual Hybrid Gradient* (SPDHG) approach, proposed by Chambolle et al. [Cha+18], is utilized. SPDHG is a stochastic extension of PDHG which randomly updates only a subset of dual variables in each iteration, in contrast to PDHG where all dual variables are updated. This randomization strategy allows SPDHG to achieve faster convergence and improved scalability for large-scale optimization problems, particularly those with a high number of dual variables.

It is worth to note that a comparably large number of dual variables naturally emerges in variational optimization problems on finite weighted graphs, where each edge in the graph introduces a new dual variable, which typically results in more dual than primal variables. This inherent property of variational problems, makes them well-suited for optimization using SPDHG.

In order to improve the runtime performance on modern computers, the original SPDHG Alg1 [Cha+18] is modified as follows. First, a *batch size* B is chosen, and the array of

dual variables is partitioned into batches, each containing B many dual variables. Then one single dual variable is randomly chosen from each batch, rather than selecting dual variables at random from the entire image as in the original approach [Cha+18]. This design choice ensures a uniform selection across dual variables. Consequently, the probability of selecting the dual variable of vertex $v \in V$ is $p_v = 1/B$. It is worth to note that if the batch size is $B = 1$, then SPDHG turns into the classical deterministic PDHG algorithm, since $p_v = 1 \forall v \in V$.

Each batch corresponds to a spatially localized region within memory where dual variables are stored. By keeping the batch size sufficiently small, this localized grouping takes advantage of spatial locality in memory access. This design enables more effective use of caching and paging mechanisms, optimizing runtime performance on modern computers by reducing memory access latency and improving cache efficiency.

As mentioned in *remark 2* by Chambolle et al. [Cha+18], the performed update of dual variables will, due to the structure of K_v , affect only a subset of primal variables. In the case of variational optimization, on a finite weighted graph these primal variables $\rho(u)$ are those which are adjacent to $v \in \mathbb{S}$, i.e. there is an edge $(u, v) \in E$. To account for this inherent property of the SPDHG algorithm, in the proposed variant the respective primal variables which are affected will be added to a set $\mathbb{S}_p \subseteq V$. In the next iteration only primal variables from this set will be updated.

The entire procedure of the *Proposed SPDHG* is shown in algorithm 6.2.

Algorithm 6.2 Proposed Stochastic PDHG (SPDHG) Algorithm

- 1: Choose $\tau, \sigma > 0$ (with $\tau\sigma\|K\|_{\text{op}}^2 \leq 1$), and relaxation parameter $\theta \in [0, 1]$
 - 2: Choose batch size B
 - 3: **repeat**
 - 4: Select $\mathbb{S}_B \subseteq V$ according to batch size B , with $p_v = \frac{1}{B}$
 - 5: **for each** $v \in \mathbb{S}_B$ **do**
 - 6: $y^{k+1}(v) = \text{prox}_{\sigma\phi_F^*}(y^k(v) + \sigma K_v \rho^k(v))$
 - 7: $\bar{y}^{k+1}(v) = y^{k+1}(v) + \frac{\theta}{p_v} (y^{k+1}(v) - y^k(v))$
 - 8: $\mathbb{S}_p = \mathbb{S}_p \cup \{v\}$, primal variable of the current vertex will be updated
 - 9: $\mathbb{S}_p = \mathbb{S}_p \cup \{u\}$, primal variables of adjacent vertices $(u, v) \in E$ will be updated
 - 10: **end for**
 - 11: **for each** $v \in \mathbb{S}_p$ **do**: $\rho^{k+1}(v) = \text{prox}_{\tau\phi_G}(\rho^k(v) - \tau K_v^\top \bar{y}^k(v))$
 - 12: **until** convergence or $k > k_{\max}$
-

This algorithm along with the convex formulation shown in section 6.3.2 are the key elements in this proposal, to speed up the convergence and runtime performance of *Cut Pursuit* (cf. section 3.2) based partitioning approaches.

6.4 EVALUATION

This section assesses the impact of the proposed enhancements from section 6.3.2 and section 6.3.4, comparing them to the classical PDHG solver in section 6.3.3. All experiments use images from the *Berkeley Segmentation Dataset* (BSDS) [Mar+01]. The focus is on the piecewise-constant Mumford-Shah problem, where ϕ_D is defined as in equation (6.1). Parameters are set to: $\lambda = 0.9$, $\theta = 1$, and $K_v = \begin{bmatrix} \nabla_w \\ [1, 1] \end{bmatrix}$ (cf. equation (6.13)), with PDHG settings $\tau = 1/\|K\|$, $\sigma = 1/\|K\|$, and likewise for SPDHG $\tau = 1/(B\|K\|)$, with K as in remark 2.20.

For quantitative analysis, the following metrics are employed [Mei05]: the Rand Index (RI), Variation of Information (VI), and Objective Error (OE). RI measures partitioning accuracy (higher values indicate better matches), VI measures information difference (lower values indicate better matches), and OE calculates the cumulative element-wise absolute difference (lower values indicate better matches). The ground truth is derived from results of the Boykov-Kolmogorov (BK) [BK03] graph cut algorithm.

Results of the quantitative ablation study is shown in Fig. 6.3. Each data point represents the average RI, VI, and OE for a specified number of iterations of different variants of (S)PDHG based graph cut algorithms. As expected, the standard PDHG method, paired with the classical formulation of the image partitioning problem (cf. equation (6.2)) exhibits the poorest performance across all metrics. Replacing the classical partitioning term with the convex formulation (cf. equation (6.11)) significantly improves the performance.

Using SPDHG with $B = 4$ and $B = 16$ alongside the classical formulation also yields improvements, achieving performance levels comparable to the PDHG+convex variant. Combining both enhancements (SPDHG+convex) achieves the best performance.

It is worth to note that a single SPDHG epoch requires more computation than a PDHG iteration (cf. table 6.1), due to a worse (stochastic) memory access pattern. Thus, while a larger batch size enhances convergence, it also demands more runtime. A batch size of $B=4$ shows a good balance between quick convergence and efficient runtime.

Approach	runtime factor
SPDHG with B=1 (PDHG)	1
SPDHG with B=4	1.5
SPDHG with B=16	2.0

Table 6.1: Runtime performance of SPDHG for different batch sizes B .

The results of the entire Cut Pursuit with PDHG+classical (2000 iterations) and the proposed SPDHG+convex approach (1000 epochs) are shown in Fig. 6.4. Despite the use of fewer iterations, the proposed method is closer to the ground truth from BK [BK03], than PDHG+classical.

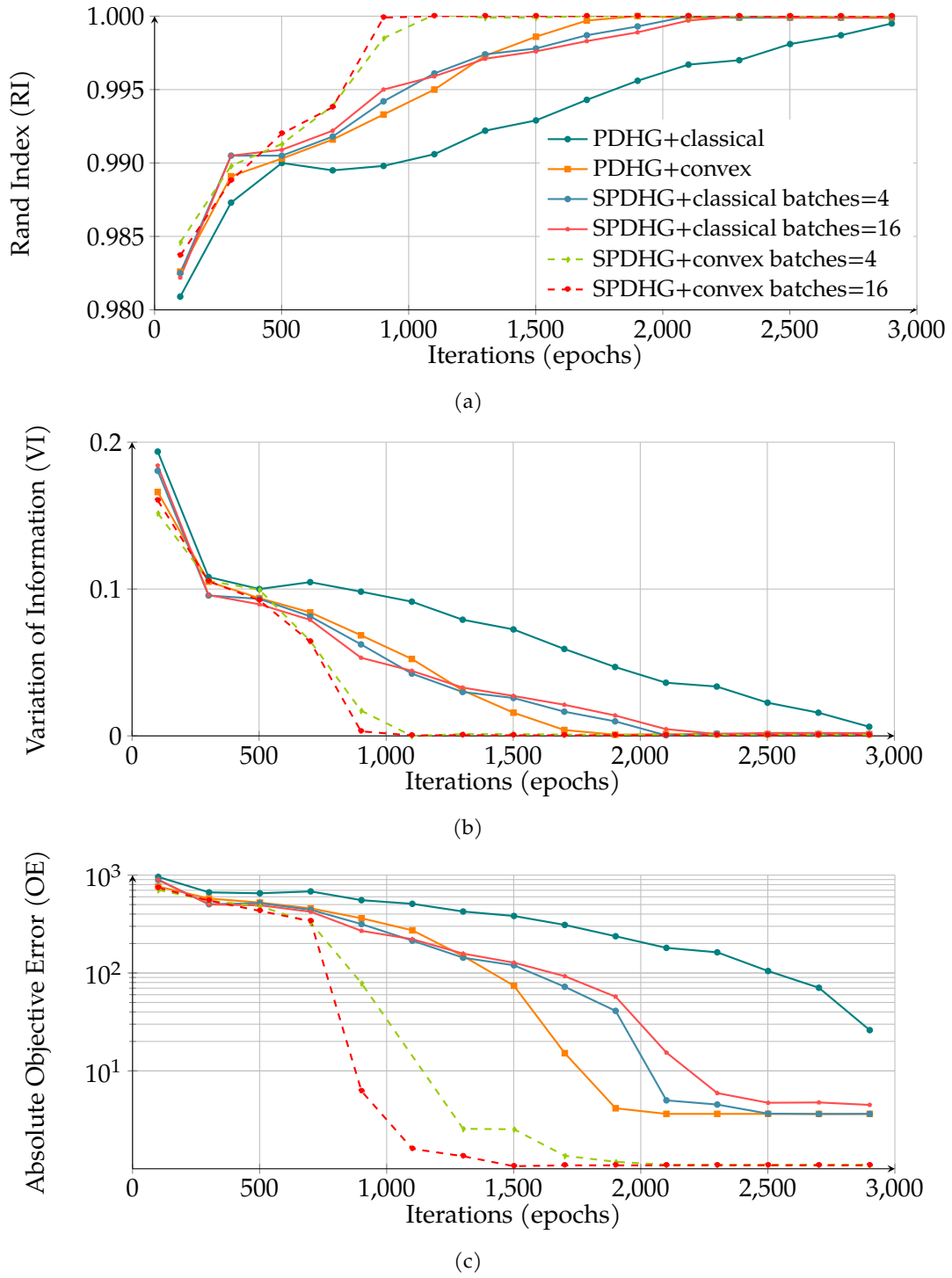


Figure 6.3: Quantitative comparison of variational graph cut results. The rand index (RI) (higher is better) is shown in (a), variation of information (VI) (lower is better) is shown in (b), and the absolute objective error (OE) (lower is better) is shown in (c). Our approach (SPDHG+convex) converges faster to favourable values with fewer iterations (epochs).

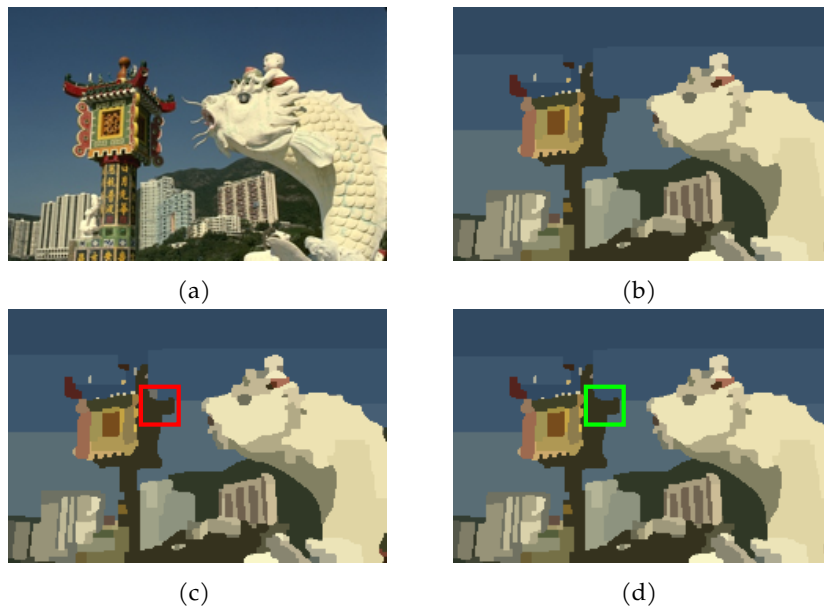


Figure 6.4: Results after five Cut Pursuit iterations. The original image (a), ground truth from BK [BK03] (b), PDHG+classical (2000 iterations, $RI=0.995$, $VI=0.04$, $OE=147$) (c) and the proposed approach (SPDHG+convex batches=4) (1000 epochs, $RI=0.999$, $VI=0.003$, $OE=2$) (d).

6.5 CONCLUSION

In this chapter a novel approach for fast estimation of piecewise constant functions based on the Cut Pursuit algorithm was proposed. The underlying graph cut algorithm was formulated as a convex variational problem, and solved via stochastic proximal gradient descent (SPDHG). The fast convergence rate of the proposed approach was highlighted by various experiments.

It is worth to note that the proposed algorithm can be extended to perform ternary partitioning and thus be applicable to non-differentiable data terms [RL18]. Furthermore this proposal does not need any changes, as in [RL19], to allow for parallelization, since SPDHG [Cha+18] is per se parallelizable and thus can be easily implemented for SIMD hardware such as GPUs.

Together with the findings of Geiping et al. [Gei+20] and Lang et al. [LSV21] the proposed approach provides an efficient algorithm for sparsification of image attributes and thus facilitates performance enhancements of energy minimization problems.

CONCLUSION

This thesis presented an analysis of the efficient and effective utilization of minimal partitioning techniques in RGB-D image processing. The proposed approaches used the L_0 “norm” together with the Moreau envelope and the Cut Pursuit algorithm to achieve better segmentation and optimization. These methods were studied in detail in chapter 3 and then applied to two different problem areas, showing their usefulness in real-world cases.

The first application focused on scene flow estimation, where the motion in 3D space was modeled as rigid transformations with a piecewise constant constraint. The constraint was formulated using L_0 “norm” regularization on the rotational and translational motion components. However, due to the high computational cost of L_0 “norm”, an alternative formulation was derived using the *truncated quadratic* “norm” along with the Moreau identity, making the optimization problem easier to solve. The clustering properties of L_0 “norm” regularization enabled an implicit segmentation of motion, reducing the need for additional pre- or post-processing. Experimental tests demonstrated that this approach provided highly accurate results while maintaining computational efficiency, making it suitable for practical use. Nevertheless, despite its advantages, the motion estimation was not strictly piecewise constant due to the inherent approximations in the algorithm. Some errors remained, particularly in cases where the rigid-motion assumption was violated. For example, in the case of elasto-plastic deformations, where the method approximated them as piecewise-rigid transformations, inaccuracies were introduced.

The second application focused on estimation of piecewise constant geometric model parameters in RGB-D images, with the goal of getting denoised and consistent depth and normal values. To achieve this, the depth data was estimated second order polynomial of inverse depth values. It was shown that this second-order model of inverse depth is equal to degenerate quadrics in 3D. The model parameters of quadrics are then used to generate dense and consistent, high-quality depth and normal maps, which preserves geometric coherence. Experimental results demonstrated that this method outperformed existing approaches in terms of accuracy and density. This was especially the case in geometric scenes with complex surface structures. The ability to produce coherent and dense depth and normal values made this approach suitable for applications, which require accurate geometry reconstruction. However, the quality of this approach comes at the cost of high computational demands. This brings challenges for e.g. real-time applications, which overall limits the practicality of this approach.

The final step of this research was to improve the runtime performance of the previous Cut Pursuit-based approach. To achieve this the underlying graph cut algorithm was formulated as a convex variational problem. This new version was then solved by using the stochastic proximal gradient descent (SPDHG) algorithm. Overall these steps improved the convergence rate and in turn led to a better runtime performance. Experimental results showed that this approach was faster than the traditional deterministic version of the algorithm. However, even though the stochastic method improved the convergence rate and reduced computation time, the runtime performance remained comparably slow. This means that further improvements are needed to make the proposed method suitable for real-time applications.

In the future, several research directions could be explored. The variational approach used in the first study highlighted the advantages of L_0 “norm” regularization but the results were not piecewise constant, as discussed. This remaining problem could be improved by using post-processing with Cut Pursuit or SLIC to enforce piecewise constant results. The Cut Pursuit-based geometric modeling approach, discussed in the second study, produced accurate and well-segmented results, but it required a comparably high amount of computational time. While stochastic proximal gradient descent sped up the process, real-time performance was still not achieved. Future research could involve using an initial fast clustering (e.g., via SLIC) followed by the Cut Pursuit approach. This could reduce the number of iterations, which would lead to an improved runtime performance. Additionally, exploring alternative optimization methods or hardware acceleration techniques could improve computational efficiency, bringing these methods closer to real-time applications. Further research could also investigate applying these techniques to other areas, leveraging their ability to effectively segment and regularize data.

BIBLIOGRAPHY

- [Ach+12] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. “SLIC Superpixels Compared to State-of-the-Art Superpixel Methods.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: [10.1109/TPAMI.2012.120](#) (cit. on p. 46).
- [AEB06] M. Aharon, M. Elad, and A. Bruckstein. “K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation.” In: *IEEE Transactions on Signal Processing* 54.11 (2006), pp. 4311–4322. DOI: [10.1109/TSP.2006.881199](#) (cit. on p. 62).
- [BCM05] A. Buades, B. Coll, and J.-M. Morel. “A non-local algorithm for image denoising.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2005, pp. 60–65. DOI: [10.1109/CVPR.2005.38](#) (cit. on p. 62).
- [Ber09] D. Bertsekas. *Convex Optimization Theory*. Vol. 1. Athena Scientific, 2009. ISBN: [1886529310](#) (cit. on pp. 12–18, 22–24).
- [BER17] B. Berkels, A. Effland, and M. Rumpf. “A Posteriori Error Control for the Binary Mumford-Shah Model.” In: *Mathematics of Computation* 86.306 (2017), pp. 1769–1791. DOI: [10.1090/mcom/3138](#) (cit. on pp. 76, 77, 79).
- [BK03] Y. Boykov and V. Kolmogorov. “Computing geodesics and minimal surfaces via graph cuts.” In: *IEEE International Conference on Computer Vision*. Vol. 1. 9. 2003, pp. 26–33. DOI: [10.1109/ICCV.2003.1238310](#) (cit. on pp. 77, 83, 85).
- [BK04] Y. Boykov and V. Kolmogorov. “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9 (2004), pp. 1124–1137. DOI: [10.1109/TPAMI.2004.60](#) (cit. on pp. 39, 76, 77).
- [BKP10] K. Bredies, K. Kunisch, and T. Pock. “Total Generalized Variation.” In: *SIAM Journal on Imaging Sciences* 3.3 (2010), pp. 492–526. DOI: [10.1137/090769521](#) (cit. on pp. 2, 61).
- [BM92] P. J. Besl and N. D. McKay. “Method for registration of 3-D shapes.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.2 (1992), pp. 239–256. DOI: [10.1109/34.121791](#) (cit. on p. 46).
- [Bro+04] T. Brox, A. Bruhn, N. Papenbergh, and J. Weickert. “High accuracy optical flow estimation based on a theory for warping.” In: *Computer Vision – ECCV 2004*. Springer, 2004, pp. 25–36. DOI: [10.1007/978-3-540-24673-2_3](#) (cit. on pp. 45, 51, 52).

-
- [BV04] S. P. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. ISBN: [0521833787](#) (cit. on pp. [12](#), [13](#), [15](#), [17](#), [18](#), [22](#), [24](#)).
 - [BVZ01] Y. Boykov, O. Veksler, and R. Zabih. “Fast approximate energy minimization via graph cuts.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11 (2001), pp. 1222–1239. DOI: [10.1109/34.969114](#) (cit. on pp. [2](#), [76](#)).
 - [CCP12] A. Chambolle, D. Cremers, and T. Pock. “A Convex Approach to Minimal Partitions.” In: *SIAM Journal on Imaging Sciences* 5.4 (2012), pp. 1113–1158. DOI: [10.1137/110856733](#) (cit. on pp. [78](#), [79](#)).
 - [CEN06] T. F. Chan, S. Esedoglu, and M. Nikolova. “Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models.” In: *SIAM Journal on Applied Mathematics* 66.5 (2006), pp. 1632–1648. DOI: [10.1137/040615286](#) (cit. on p. [67](#)).
 - [Cha+18] A. Chambolle, M. J. Ehrhardt, P. Richtárik, and C.-B. Schonlieb. “Stochastic Primal-Dual Hybrid Gradient Algorithm with Arbitrary Sampling and Imaging Applications.” In: *SIAM Journal on Optimization* 28.4 (2018), pp. 2783–2808. DOI: [10.1137/17M1134834](#) (cit. on pp. [76](#), [77](#), [81](#), [82](#), [85](#)).
 - [CL97] A. Chambolle and P.-L. Lions. “Image recovery via total variation minimization and related problems.” In: *Numerische Mathematik* 76.2 (1997), pp. 167–188. DOI: [10.1007/s002110050258](#) (cit. on p. [61](#)).
 - [CM91] Y. Chen and G. Medioni. “Object Modeling by Registration of Multiple Range Images.” In: *IEEE International Conference on Robotics and Automation* 10.3 (1991), pp. 2724–2729. DOI: [10.1109/ROBOT.1991.132043](#) (cit. on p. [46](#)).
 - [CP11] A. Chambolle and T. Pock. “A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging.” In: *Journal of Mathematical Imaging and Vision* 40.1 (2011), pp. 120–145. DOI: [10.1007/s10851-010-0251-1](#) (cit. on pp. [7](#), [28](#), [29](#), [31](#), [37](#), [44](#)).
 - [CP16] A. Chambolle and T. Pock. “An introduction to continuous optimization for imaging.” In: *Acta Numerica* 25 (2016), pp. 161–319. DOI: [10.1017/S096249291600009X](#) (cit. on pp. [2](#), [22](#), [33](#), [39](#)).
 - [Dab+07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian. “Image Denoising by Sparse 3-D Transform-Domain Collaborative Filtering.” In: *IEEE Transactions on Image Processing* 16.8 (2007), pp. 2080–2095. DOI: [10.1109/TIP.2007.901238](#) (cit. on p. [62](#)).
 - [DB06] H. Durrant-Whyte and T. Bailey. “Simultaneous localization and mapping: part I.” In: *IEEE Robotics Automation Magazine* 13.2 (2006), pp. 99–110. DOI: [10.1109/MRA.2006.1638022](#) (cit. on p. [43](#)).
 - [Fer+13] D. Ferstl, C. Reinbacher, R. Ranftl, M. Ruether, and H. Bischof. “Image Guided Depth Upsampling Using Anisotropic Total Generalized Variation.” In: *2013 IEEE International Conference on Computer Vision*. 2013, pp. 993–1000. DOI: [10.1109/ICCV.2013.127](#) (cit. on pp. [2](#), [61](#)).
-

- [FTK14] C. Feng, Y. Taguchi, and V. R. Kamat. “Fast plane extraction in organized point clouds using agglomerative hierarchical clustering.” In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 6218–6225. DOI: [10.1109/ICRA.2014.6907776](#) (cit. on pp. [2](#), [60](#), [62](#)).
- [Gei+20] J. Geiping, F. Gaede, H. Bauermeister, and M. Moeller. “Fast Convex Relaxations using Graph Discretizations.” In: *31st British Machine Vision Conference (BMVC)*. 2020. URL: <https://www.bmvc2020-conference.com/assets/papers/0694.pdf> (cit. on pp. [1](#), [62](#), [75](#), [76](#), [85](#)).
- [GGK19] A. Görlitz, J. Geiping, and A. Kolb. “Piecewise Rigid Scene Flow with Implicit Motion Segmentation.” In: *IEEE International Conference on Intelligent Robots and Systems*. 2019, pp. 1758–1765. DOI: [10.1109/IR0540897.2019.8968018](#) (cit. on pp. [4](#), [43](#)).
- [GH97] M. Garland and P. S. Heckbert. “Surface Simplification using Quadric Error Metrics.” In: *SIGGRAPH*. ACM, 1997, pp. 209–216. DOI: [10.1145/258734.258849](#) (cit. on p. [68](#)).
- [Ghu+14] S. Ghuffar, N. Brosch, N. Pfeifer, and M. Gelautz. “Motion estimation and segmentation in depth and intensity videos.” In: *Integrated Computer-Aided Engineering* 21.3 (2014), pp. 203–218. DOI: [10.3233/ICA-130456](#) (cit. on p. [45](#)).
- [GMK22] A. Görlitz, M. Möller, and A. Kolb. “FL0C: Fast L0 Cut Pursuit for Estimation of Piecewise Constant Functions.” In: *IEEE International Conference on Image Processing*. 2022, pp. 3677–3681. DOI: [10.1109/ICIP46576.2022.9897892](#) (cit. on pp. [4](#), [75](#)).
- [GMK24] A. Görlitz, M. Möller, and A. Kolb. “Coherent Enhancement of Depth Images and Normal Maps Using Second-Order Geometric Models on Weighted Finite Graphs.” In: *International Conference on 3D Vision*. 2024, pp. 623–630. DOI: [10.1109/3DV62453.2024.00023](#) (cit. on pp. [4](#), [59](#)).
- [Gol+17] V. Golyanik, K. Kim, R. Maier, M. Nießner, D. Stricker, and J. Kautz. “Multi-frame scene flow with piecewise rigid motion.” In: *International Conference on 3D Vision*. 2017, pp. 273–281. DOI: [10.1109/3DV.2017.00039](#) (cit. on pp. [44](#), [46](#)).
- [Grz+13] M. Grzegorzec, C. Theobalt, R. Koch, and A. Kolb. *Time-of-Flight and Depth Imaging. Sensors, Algorithms and Applications: Dagstuhl Seminar 2012 and GCPR Workshop on Imaging New Modalities*. Vol. 8200. Springer, 2013. ISBN: [3642449638](#) (cit. on p. [7](#)).
- [GSC13] B. Goldlücke, E. Strekalovskiy, and D. Cremers. “Tight Convex Relaxations for Vector-Valued Labeling.” In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1626–1664. DOI: [10.1137/120862351](#) (cit. on p. [76](#)).
- [Han+12] M. Hansard, S. Lee, O. Choi, and R. P. Horaud. *Time-of-Flight Cameras: Principles, Methods and Applications*. Springer, 2012. DOI: [10.1007/978-1-4471-4658-2](#) (cit. on p. [7](#)).
-

- [Han+14] A. Handa, T. Whelan, J. McDonald, and A. J. Davison. “A benchmark for RGB-D visual odometry, 3D reconstruction and SLAM.” In: *IEEE International Conference on Robotics and Automation*. 2014, pp. 1524–1531. DOI: [10.1109/ICRA.2014.6907054](#) (cit. on pp. [60](#), [69–71](#)).
- [HM99] J. Huang and D. Mumford. “Statistics of natural images and models.” In: *IEEE Computer Conference on Computer Vision and Pattern Recognition*. Vol. 1. 1999, pp. 541–547. DOI: [10.1109/CVPR.1999.786990](#) (cit. on p. [61](#)).
- [HRF13] E. Herbst, X. Ren, and D. Fox. “RGB-D flow: Dense 3-D motion estimation using color and depth.” In: *IEEE International Conference on Robotics and Automation*. 2013, pp. 2276–2282. DOI: [10.1109/ICRA.2013.6630885](#) (cit. on pp. [45](#), [47](#)).
- [Hub64] P. J. Huber. “Robust Estimation of a Location Parameter.” In: *The Annals of Mathematical Statistics* 35.1 (1964), pp. 73–101. DOI: [10.1214/aoms/1177703732](#) (cit. on p. [67](#)).
- [HW13] M. Hintermüller and T. Wu. “Nonconvex TV^q -Models in Image Restoration: Analysis and a Trust-Region Regularization–Based Superlinearly Convergent Solver.” In: *SIAM Journal on Imaging Sciences* 6.3 (2013), pp. 1385–1415. DOI: [10.1137/110854746](#) (cit. on p. [61](#)).
- [HZ00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. ISBN: [0521623049](#) (cit. on pp. [7](#), [64](#)).
- [Inn+16] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, and M. Stamminger. “VolumeDeform: Real-Time Volumetric Non-rigid Reconstruction.” In: *Computer Vision – ECCV 2016*. Springer. 2016, pp. 362–379. DOI: [10.1007/978-3-319-46484-8_22](#) (cit. on p. [46](#)).
- [Isi24] E. Ising. “Beitrag zur Theorie des Ferromagnetismus.” In: *Zeitschrift für Physik* 31.1 (1924), pp. 253–258. DOI: [10.1007/BF02980577](#) (cit. on p. [33](#)).
- [Jai+15] M. Jaimez, M. Souiai, J. Gonzalez-Jimenez, and D. Cremers. “A primal-dual framework for real-time dense RGB-D scene flow.” In: *IEEE International Conference on Robotics and Automation*. 2015, pp. 98–104. DOI: [10.1109/ICRA.2015.7138986](#) (cit. on pp. [44](#), [45](#), [55–57](#)).
- [Jai+17] M. Jaimez, C. Kerl, J. Gonzalez-Jimenez, and D. Cremers. “Fast odometry and scene flow from RGB-D cameras based on geometric clustering.” In: *IEEE International Conference on Robotics and Automation*. 2017, pp. 3992–3999. DOI: [10.1109/ICRA.2017.7989459](#) (cit. on pp. [44](#), [46](#), [55–57](#)).
- [Kap+15] J. H. Kappes et al. “A Comparative Study of Modern Inference Techniques for Structured Discrete Energy Minimization Problems.” In: *International Journal of Computer Vision* 115.2 (2015), pp. 155–184. DOI: [10.1007/s11263-015-0809-x](#) (cit. on p. [76](#)).
- [Kel75] J. L. Kelley. *General Topology*. Vol. 27. Graduate Texts in Mathematics. Springer, 1975. DOI: [10.2307/2964144](#) (cit. on pp. [12](#), [13](#)).

- [Kol+10] A. Kolb, E. Barth, R. Koch, and R. Larsen. "Time-of-Flight Cameras in Computer Graphics." In: *Computer Graphics Forum* 29.1 (2010), pp. 141–159. DOI: [10.1111/j.1467-8659.2009.01583.x](#) (cit. on p. 7).
- [Kor+22] A. Kornilova, D. Iarosh, D. Kukushkin, N. Goncharov, P. Mokeev, A. Saliou, and G. Ferrer. "EVOPS Benchmark: Evaluation of Plane Segmentation from RGBD and LiDAR Data." In: *IEEE International Conference on Intelligent Robots and Systems*. 2022, pp. 13074–13080. DOI: [10.1109/IRoS47612.2022.9981470](#) (cit. on pp. 60, 69).
- [KSW20] L. Kiefer, M. Storath, and A. Weinmann. "PALMS Image Partitioning - A New Parallel Algorithm for the Piecewise Affine-Linear Mumford-Shah Model." In: *Image Processing On Line* 10 (2020), pp. 124–149. DOI: [10.5201/ipol.2020.295](#) (cit. on pp. 61, 69–71, 74).
- [Llo82] S. Lloyd. "Least Squares Quantization in PCM." In: *IEEE Transactions on Information Theory* 28.2 (1982), pp. 129–137. DOI: [10.1109/TIT.1982.1056489](#) (cit. on p. 55).
- [LO17] L. Landrieu and G. Obozinski. "Cut Pursuit: Fast Algorithms to Learn Piecewise Constant Functions on General Weighted Graphs." In: *SIAM Journal on Imaging Sciences* 10.4 (2017), pp. 1724–1766. DOI: [10.1137/17M1113436](#) (cit. on pp. 38, 39, 66, 68, 76–78).
- [Lou+15] Y. Lou, P. Yin, Q. He, and J. Xin. "Computing Sparse Representation in a Highly Coherent Dictionary Based on Difference of L1 and L2." In: *ACM Journal of Scientific Computing* 64.1 (2015), pp. 178–196. DOI: [10.1007/s10915-014-9930-1](#) (cit. on p. 2).
- [LSV21] H. Lang, D. A. Sontag, and A. Vijayaraghavan. "Graph Cuts Always Find a Global Optimum for Potts Models (With a Catch)." In: *Proceedings of the International Conference on Machine Learning*. Vol. 139. 2021, pp. 5990–5999. DOI: [10.48550/arXiv.2011.03639](#) (cit. on pp. 76, 85).
- [LZP14] L. Ladický, B. Zeisl, and M. Pollefeys. "Discriminatively trained dense surface normal estimation." In: *Computer Vision – ECCV 2014*. 2014, pp. 468–484. DOI: [10.1007/978-3-319-10602-1_31](#) (cit. on pp. 69, 73).
- [Mah36] P. C. Mahalanobis. "On the generalized distance in statistics." In: *Proceedings of the National Institute of Sciences* 2 (1936), pp. 49–55. DOI: [10.1007/s13171-019-00164-5](#) (cit. on p. 68).
- [Mar+01] D. Martin, C. Fowlkes, D. Tal, and J. Malik. "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics." In: *IEEE International Conference on Computer Vision*. Vol. 2. 2001, pp. 416–423. DOI: [10.1109/ICCV.2001.937655](#) (cit. on p. 83).
- [Meh+21] N. Mehrabi, F. Morstatter, N. Saxena, K. Lerman, and A. Galstyan. "A Survey on Bias and Fairness in Machine Learning." In: *ACM Computing Surveys* 54.6 (2021), 115:1–115:35. DOI: [10.1145/3457607](#) (cit. on p. 61).

- [Mei05] M. Meila. “Comparing Clusterings: An Axiomatic View.” In: *Proceedings of the International Conference on Machine Learning*. 2005, pp. 577–584. DOI: [10.1145/1102351.1102424](#) (cit. on p. [83](#)).
- [Möl+16] T. Möllenhoff, E. Laude, M. Möller, J. Lellmann, and D. Cremers. “Sublabel-Accurate Relaxation of Nonconvex Energies.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3948–3956. DOI: [10.1109/CVPR.2016.428](#) (cit. on p. [55](#)).
- [MS89] D. Mumford and J. Shah. “Optimal approximations by piecewise smooth functions and associated variational problems.” In: *Communications on Pure and Applied Mathematics* 42.5 (1989), pp. 577–685. DOI: [10.1002/cpa.3160420503](#) (cit. on pp. [33](#), [61](#), [76](#)).
- [NB15] R. M. H. Nguyen and M. S. Brown. “Fast and Effective L0 Gradient Minimization by Region Fusion.” In: *IEEE International Conference on Computer Vision*. 2015, pp. 208–216. DOI: [10.1109/ICCV.2015.32](#) (cit. on p. [1](#)).
- [NTC13] C. Nieuwenhuis, E. Töppe, and D. Cremers. “A Survey and Comparison of Discrete and Continuous Multi-label Optimization Approaches for the Potts Model.” In: *International Journal of Computer Vision* 104.3 (2013), pp. 223–240. DOI: [10.1007/s11263-013-0619-y](#) (cit. on p. [76](#)).
- [PB14] N. Parikh and S. Boyd. “Proximal Algorithms.” In: *Foundations and Trends in Optimization* 1.3 (2014), pp. 127–239. DOI: [10.1561/24000000003](#) (cit. on pp. [22](#), [24–27](#)).
- [PC11] T. Pock and A. Chambolle. “Diagonal preconditioning for first order primal-dual algorithms in convex optimization.” In: *International Conference on Computer Vision*. 2011, pp. 1762–1769. DOI: [10.1109/ICCV.2011.6126441](#) (cit. on pp. [29](#), [30](#)).
- [PG18] P. F. Proença and Y. Gao. “Fast Cylinder and Plane Extraction from Depth Cameras for Visual Odometry.” In: *IEEE International Conference on Intelligent Robots and Systems*. 2018, pp. 6813–6820. DOI: [10.1109/IRoS.2018.8593516](#) (cit. on pp. [2](#), [59](#), [60](#), [62](#), [69](#), [70](#), [74](#)).
- [Poc+08] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers. “A Convex Formulation of Continuous Multi-Label Problems.” In: *Computer Vision – ECCV 2008*. Springer, 2008. DOI: [10.1007/978-3-540-88690-7_59](#) (cit. on p. [76](#)).
- [Poc+10] T. Pock, D. Cremers, H. Bischof, and A. Chambolle. “Global Solutions of Variational Models with Convex Regularization.” In: *SIAM Journal on Imaging Sciences* 3.4 (2010), pp. 1122–1145. DOI: [10.1137/090757617](#) (cit. on pp. [62](#), [76](#)).
- [Pot52] R. B. Potts. “Some generalized order-disorder transformations.” In: *Mathematical Proceedings of the Cambridge Philosophical Society* 48.1 (1952), pp. 106–109. DOI: [10.1017/S0305004100027419](#) (cit. on p. [33](#)).

- [QDC13] J. Quiroga, F. Devernay, and J. Crowley. “Local/global scene flow estimation.” In: *IEEE International Conference on Image Processing*. 2013, pp. 3850–3854. DOI: [10.1109/ICIP.2013.6738793](#) (cit. on p. [45](#)).
- [Qui+14] J. Quiroga, T. Brox, F. Devernay, and J. Crowley. “Dense semi-rigid scene flow estimation from RGBD images.” In: *Computer Vision – ECCV 2014*. Springer. 2014, pp. 567–582. DOI: [10.1007/978-3-319-10584-0_37](#) (cit. on pp. [33](#), [44](#), [45](#), [55–57](#)).
- [RA17] M. Rünz and L. Agapito. “Co-Fusion: Real-time Segmentation, Tracking and Fusion of Multiple Objects.” In: *IEEE International Conference on Robotics and Automation*. 2017, pp. 4471–4478. DOI: [10.1109/ICRA.2017.7989518](#) (cit. on pp. [44](#), [45](#)).
- [RL18] H. Raguét and L. Landrieu. “Cut-Pursuit Algorithm for Regularizing Non-smooth Functionals with Graph Total Variation.” In: *Proceedings of the International Conference on Machine Learning*. Vol. 80. 2018, pp. 4247–4256. DOI: [10.48550/arXiv.1802.04383](#) (cit. on p. [85](#)).
- [RL19] H. Raguét and L. Landrieu. “Parallel Cut Pursuit for Minimization of the Graph Total Variation.” In: *Computing Research Repository* (2019). DOI: [10.48550/arXiv.1905.02316](#) (cit. on pp. [76](#), [85](#)).
- [RMB21] A. Roychoudhury, M. Missura, and M. Bennewitz. “Plane Segmentation Using Depth-Dependent Flood Fill.” In: *IEEE International Conference on Intelligent Robots and Systems*. 2021, pp. 2210–2216. DOI: [10.1109/IR0551168.2021.9635930](#) (cit. on pp. [2](#), [60](#), [62](#), [69](#), [70](#), [74](#)).
- [Roc70] R. T. Rockafellar. *Convex Analysis*. Princeton University Press, 1970. ISBN: [9780691080697](#) (cit. on pp. [12](#), [13](#), [16](#), [22](#)).
- [Rod40] O. Rodrigues. “Des lois geometriques qui regissent les déplacements d’un systeme solide dans l’espace et de la variation des coordonnees provenant de déplacements consideres independamment des causes qui peuvent les produire.” In: *Journal de Mathématiques Pures et Appliquées* 1.5 (1840), pp. 380–440. ISSN: [0021-7874](#) (cit. on p. [49](#)).
- [ROF92] L. I. Rudin, S. Osher, and E. Fatemi. “Nonlinear total variation based noise removal algorithms.” In: *Physica D: Nonlinear Phenomena* 60.1 (1992), pp. 259–268. DOI: [10.1016/0167-2789\(92\)90242-F](#) (cit. on pp. [2](#), [33](#), [61](#), [76](#)).
- [Ros+20] M. Rossi, M. E. Gheche, A. Kuhn, and P. Frossard. “Joint Graph-Based Depth Refinement and Normal Estimation.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2020, pp. 12151–12160. DOI: [10.1109/CVPR42600.2020.01217](#) (cit. on pp. [61](#), [62](#), [64](#), [69–72](#), [74](#)).
- [SA07] O. Sorkine and M. Alexa. “As-rigid-as-possible surface modeling.” In: *Proceedings of the Fifth Eurographics Symposium on Geometry Processing*. Vol. 4. 2007, pp. 109–116. DOI: [10.5555/1281991.1282006](#) (cit. on p. [46](#)).
-

- [SB15] J. Stückler and S. Behnke. “Efficient Dense Rigid-Body Motion Segmentation and Estimation in RGB-D Video.” In: *International Journal of Computer Vision* 113.3 (2015), pp. 233–245. DOI: [10.1007/s11263-014-0796-3](#) (cit. on pp. [44](#), [55](#)).
- [SC14] E. Strekalovskiy and D. Cremers. “Real-Time Minimization of the Piecewise Smooth Mumford-Shah Functional.” In: *Computer Vision – ECCV 2014*. Springer, 2014, pp. 127–141. DOI: [10.1007/978-3-319-10605-2_9](#) (cit. on pp. [2](#), [34](#), [36](#), [37](#), [44](#)).
- [Sil+12] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. “Indoor Segmentation and Support Inference from RGBD Images.” In: *Computer Vision – ECCV 2012*. Springer, 2012, pp. 746–760. DOI: [10.1007/978-3-642-33715-4_54](#) (cit. on p. [69](#)).
- [Sla+17] M. Slavcheva, M. Baust, D. Cremers, and S. Ilic. “KillingFusion: Non-rigid 3D Reconstruction without espondences.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 5474–5483. DOI: [10.1109/CVPR.2017.581](#) (cit. on p. [46](#)).
- [SLK15] H. Sarbolandi, D. Lefloch, and A. Kolb. “Kinect range sensing: Structured-light versus Time-of-Flight Kinect.” In: *Computer Vision and Image Understanding* 139 (2015), pp. 1–20. ISSN: [1077-3142](#). DOI: [10.1016/j.cviu.2015.05.006](#) (cit. on pp. [2](#), [7](#), [59](#)).
- [Sol+11] J. Solomon, M. Ben-Chen, A. Butscher, and L. Guibas. “As-Killing-As-Possible Vector Fields for Planar Deformation.” In: *Computer Graphics Forum* 30.5 (2011), pp. 1543–1552. DOI: [10.1111/j.1467-8659.2011.02028.x](#) (cit. on p. [46](#)).
- [SSP15] D. Sun, E. B. Sudderth, and H. Pfister. “Layered RGBD scene flow estimation.” In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 548–556. DOI: [10.1109/CVPR.2015.7298653](#) (cit. on p. [46](#)).
- [SSW15] Y. Sun, S. Schaefer, and W. Wang. “Denoising Point Sets via L0 Minimization.” In: *Computer Aided Geometric Design* 35-36 (2015), pp. 2–15. DOI: [10.1016/j.cagd.2015.03.011](#) (cit. on p. [1](#)).
- [Stu+12] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers. “A benchmark for the evaluation of RGB-D SLAM systems.” In: *IEEE International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580. DOI: [10.1109/IR0S.2012.6385773](#) (cit. on pp. [44](#), [55](#), [57](#)).
- [TGB19] D. Tenbrinck, F. Gaede, and M. Burger. “Variational Graph Methods for Efficient Point Cloud Sparsification.” In: *math.NA* (2019). DOI: [10.48550/arXiv.1903.02858](#) (cit. on pp. [1](#), [21](#), [66](#), [68](#), [76](#), [78](#)).
- [TM98] C. Tomasi and R. Manduchi. “Bilateral filtering for gray and color images.” In: *IEEE International Conference on Computer Vision*. 1998, pp. 839–846. DOI: [10.1109/ICCV.1998.710815](#) (cit. on pp. [2](#), [61](#)).

- [Ved+05] S. Vedula, P. Rander, R. T. Collins, and T. Kanade. “Three-dimensional scene flow.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.3 (2005), pp. 475–480. DOI: [10.1109/TPAMI.2005.63](#) (cit. on p. [45](#)).
- [WYZ19] Y. Wang, W. Yin, and J. Zeng. “Global Convergence of ADMM in Non-convex Nonsmooth Optimization.” In: *Journal of Scientific Computing* 78.1 (2019), pp. 29–63. DOI: [10.1007/s10915-018-0757-z](#) (cit. on p. [53](#)).
- [Ye+21] Z. Ye, B. Haefner, Y. Quéau, T. Möllenhoff, and D. Cremers. “Sublabel-Accurate Multilabeling Meets Product Label Spaces.” In: *German Conference on Pattern Recognition*. 2021, pp. 3–17. DOI: [10.1007/978-3-030-92659-5_1](#) (cit. on p. [76](#)).
- [YX17] Z. Yan and X. Xiang. “Scene Flow Estimation: A Survey.” In: *Computing Research Repository* (2017). DOI: [10.48550/arXiv.1612.02590](#) (cit. on pp. [44](#), [45](#)).
- [Zan+18] P. Zanuttigh, G. Marin, C. D. Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo. *Time-of-Flight and Structured Light Depth Cameras: Technology and Applications*. 1st. Springer, 2018. ISBN: [3319809334](#) (cit. on p. [7](#)).
- [Zha00] Z. Zhang. “A flexible new technique for camera calibration.” In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22.11 (2000), pp. 1330–1334. DOI: [10.1109/34.888718](#) (cit. on p. [11](#)).
- [Zol+18] M. Zollhöfer, P. Stotko, A. Görlitz, C. Theobalt, M. Nießner, R. Klein, and A. Kolb. “State of the Art on 3D Reconstruction with RGB-D Cameras.” In: *Computer Graphics Forum* 37.2 (2018), pp. 625–652. DOI: [10.1111/cgf.13386](#) (cit. on pp. [1](#), [4](#), [43](#), [45](#), [59–61](#)).
-