Gottfried Munda

Dense Image Matching

DOCTORAL THESIS

to achieve the university degree of
Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Prof. Dr. Thomas Pock
Institute for Computer Graphics and Vision
Graz University of Technology, Austria

Prof. Dr. Peter Ochs
Department of Mathematics
Saarland University, Germany

Graz, Austria, 2018

To Anna

The most exciting phrase in science is not “Eureka!” but “That’s funny…”

Isaac Asimov (1920 - 1992)

Abstract

In this thesis we consider the dense image matching problem, where the goal is to find for every pixel its corresponding match in a sequence of images. Determining correspondence is among the most fundamental low-level problems in computer vision, and dense correspondences appear as an integral part of many higher-level applications. Being inherently ill-posed, the dense matching problem is typically tackled by an energy minimization consisting of a data term, which models the deviation of the solution from the data, and a regularization term, which imposes prior knowledge on the solution.

We propose to improve the modeling power of the regularization term by incorporating the geometry of the solution manifold. Using techniques from differential geometry, we formulate a physically meaningful regularizer based on the differential surface area element. We show how the differential area element can be pulled back under orthographic and perspective projection, resulting in a regularization functional on the image plane that regularizes a physically meaningful property of the surface. Applying the idea to the dense stereo problem, we show how the non-convex area functional can be convexified by means of a suitable reparameterization. We perform a number of experiments to show that area regularization has no bias towards fronto-parallel surfaces and results in higher-quality depthmaps compared to Total Variation (TV) regularization. Next, we apply the idea of manifold regularization to the problem of intensity image reconstruction for neuromorphic cameras. Such cameras do not operate on a frame basis, but instead deliver a continuous stream of events indicating brightness changes for the individual pixels. We formulate the reconstruction of grayscale images from these events as a variational problem, which is defined on a manifold given by the timestamps of the events. Our model uses the geometry of the manifold to guide the regularization, and we show that this approach results in more pronounced edges and higher contrast of the reconstructed intensity images.

Turning to the data term of the dense matching energy, we consider the problem of learning descriptors for optical flow by means of a Convolutional Neural Network (CNN). Whereas the learning of descriptors based on a multiclass classification model has shown excellent results in the context of stereo matching, applying such model to optical flow is intractable due to the quadratic memory complexity. We propose a dimensionality reduction via min-projection of the four-dimensional optical flow cost function, which reduces the memory complexity from quadratic to linear. Moreover, we accelerate the computation by using binary features. Learning of binary *CNNs* is challenging, because the hard nonlinearity used in the binarization step results in zero gradient almost everywhere. Previous state of the art circumvented this problem through the use of the so-called straight through estimation of gradients, which in effect simply discards the hard nonlinearity during gradient computation. We propose a novel hybrid learning scheme, which in the context of learning descriptors for matching significantly improves upon the straight through estimator. In the Conditional Random Field (CRF)-inference step, we apply the concept of dimensionality reduction to decompose the graphical model into a series of subproblems. We use a massively parallel solver to compute solutions of the dual of the decomposition, and we show that the inter-plane updates correspond to a min-projection with additional offsets. Our approach enables *CRF* inference on high resolution images using the full quadratic label space at linear memory complexity.

Keywords. Correspondence, Stereo, Optical Flow, Variational Methods, Graphical Models, Differential Geometry

Kurzfassung

Diese Arbeit beschäftigt sich mit dem dichten Korrespondenzproblem, bei dem das Ziel darin besteht für jeden Pixel den entsprechenden Partner in einer Sequenz von Bildern zu finden. Das Korrespondenzproblem gehört zu den grundlegendsten Problemen des maschinellen Sehens und tritt als wichtiger Teil in vielen Anwendungen auf. Da die Aufgabenstellung mathematisch schlecht gestellt ist, wird das Problem als Energieminimierung formuliert. Die Energie besteht aus einem Datenterm, welcher die Abweichung der Lösung zu den Daten misst, sowie einem Regularisierungsterm, welcher die a-priori Annahmen modelliert.

Wir zeigen, wie die Modellierungskraft des Regularisierungsterms erhöht werden kann, indem die Geometrie der Lösungsmannigfaltigkeit berücksichtigt wird. Mit Hilfe von Techniken der Differentialgeometrie formulieren wir einen physikalisch sinnvollen Regularisierer basierend auf dem differentiellen Oberflächenelement. Wir zeigen, wie das differentielle Oberflächenelement entlang der orthographischen und perspektivischen Projektion zurückgezogen werden kann. Das REsultat ist ein Regularisierungsfunktional, welches auf der Bildebene definiert ist, jedoch eine physikalisch sinnvolle Eigenschaft der Oberfläche regularisiert. Wir wenden diese Idee auf das dichte Stereo Problem an, und zeigen wie das nicht konvexe Oberflächenfunktional durch eine passende Reparametrisierung in ein konvexes Funktional transformiert werden kann. Eine Reihe von Experimenten bestätigt, dass das Oberflächenfunktional im Unterschied zu TV-Regularisierung keinen systematischen Bias zu stückweise konstanten Funktionen aufweist, wodurch wir Tiefenmaps von höherer Qualität erhalten. Darüber hinaus setzen wir die Idee der Mannigfaltigkeits-Regularisierung im Bildrekonstruktionsproblem für Event Kameras ein. Solche Kameras erzeugen kein gewöhnliches Bild, sondern einen kontinuierlichen Strom von Events basierend auf der Helligkeitsänderung der einzelnen Pixel. Wir formulieren die Rekonstruktion von Graustufenbildern aus den Events als Variationsproblem auf einer Mannigfaltigkeit, welche durch die Zeitstempel der Events gegeben ist. Unser Modell nutzt die Geometrie der Mannigfaltigkeit als Leitlinie für die Regularisierung. Wir zeigen, dass dieser Ansatz bessere Kanten und höheren Kontrast in den rekonstruierten Bildern ergibt.

Als nächstes beschäftigen wir uns mit dem Lernen von Deskriptoren für optischen Fluss mit Hilfe eines *CNN*. Lernansätze basierend auf dem Mehr-Klassen Klassifikationsmodell zeigten bereits sehr gute Ergebnisse im Stereo Problem, sind jedoch wegen der quadratischen Speicherkomplexität für optischen Fluss nicht direkt anwendbar. Wir führen eine Minimum-Projektion der vierdimensionalen Kostenfunktion ein, welche die Speicherkomplexität von quadratisch auf linear reduziert. Darüber hinaus beschleunigen wir die Berechnungen durch Verwendung von binären Deskriptoren. Das Lernen von binären *CNNs* ist schwierig, da der Gradient wegen der harten Nichtlinearität, welche für die Binarisierung verwendet wird, fast überall Null ist. Frühere Arbeiten umgingen dieses Problem durch den sogenannten straight-through Schätzer für den Gradienten, welcher bei Berechnung des Gradienten die Nichtlinearität vernachlässigt. Wir führen eine neuartige hybride Lernstrategie ein, welche die Resultate im Vergleich zum straight-through Schätzer signifikant verbessert. Im *CRF*-Inferenz Schritt zerlegen wir den Berechnungsgraphen in eine Reihe von Unterproblemen, was zu einer Komplexitätsreduktion führt. Um die Dualprobleme innerhalb der Ebenen zu lösen, verwenden wir einen massiv parallelen Algorithmus, und wir zeigen dass die Update Schritte zwischen den Ebenen einer Minimum-Projektion mit zusätzlichen Kostenoffsets entsprechen. Unser Ansatz ermöglicht effiziente *CRF*-Inferenz auf hochauflösenden Bildern mit dem vollen quadratischen Merkmalsraum bei linearer Speicherkomplexität.

Schlagwörter. Korrespondenz, Stereo, Optischer Fluss, Variationsmethoden, Probabilistische Graphische Modelle, Differentialgeometrie

Acknowledgments

I would like to take the opportunity and thank all the people I have met during my years of studying and who had an impact on this thesis. First and foremost I am greatly indebted to my supervisor Thomas Pock for his knowledge and guidance during my PhD. Working in his group at the ICG is an experience I will hold dear forever. I also thank Peter Ochs for agreeing to be my second supervisor.

I would like to thank the colleagues who shared their knowledge with me in many fruitful discussions, especially René Ranftl, Stefan Heber, Christian Reinbacher, Alexander Shekhovtsov, Christoph Vogel, Erich Kobler, Patrick Knöbelreiter and all other former and current members of the VLO group. It was a pleasure learning in such an exciting environment. My thanks go to Stefano Soatto, who agreed to take me as exchange researcher at UCLA twice. During these visits I met Jonathan Balzer, with whom I developed the ideas for the manifold regularization which became an important part of this thesis.

Finally I am thankful to my family and my friends, and especially to Anna for her understanding, support and love.

Contents

List of Figures

List of Tables

1 Introduction

Contents

1.1 The Correspondence Problem

The goal of computer vision algorithms is to infer higher level information from images. It has been recognized since the early beginnings of computer vision that the ability to find correspondences in images is very useful for this task. Indeed, as soon as we have a collection of images that in some sense "show the same"[1], the concept of correspondences arises naturally. It is therefore no surprise that the search for correspondences is an integral part of many machine vision algorithms, with applications ranging from stereo and optical flow to medical image registration, face recognition, visual odometry, tracking and 3D-reconstruction. For example, if we have an image sequence depicting moving objects, correspondences allow to *track* the objects. On the other hand, if we have images of a static scene taken from different viewpoints, correspondences allow to *reconstruct* 3D structure.

We define the correspondence problem informally as follows:

Given a point in one image, find the corresponding point in the other image.

Let us point out two difficulties:

[1]Images of the scene taken at/from a) different vantage points and same time, b) same vantage points and different time, c) different vantage points and different time.

a) It is not clear how to quantify and measure “correspondence”. One intuitive idea is to measure differences of intensity values. However, in case of appearance changes this breaks down: corresponding points could have an arbitrarily different pixel value.

b) Sometimes a corresponding point might not exist at all. Moving objects and varying viewpoints cause occlusions, i.e. parts of one image that are not visible in the other image.

The underlying root cause of problems a) and b) is that correspondences are generated by the projection of a 3D point onto the 2D image plane. Unfortunately, the same 3D point can project to quite different image points, see fig. 1.1. If the images are taken at different times, lighting conditions and shadows may have changed, which results in problem a). Even if the images are taken at the same time, varying vantage points and moving objects will result in occlusions and thus problem b).

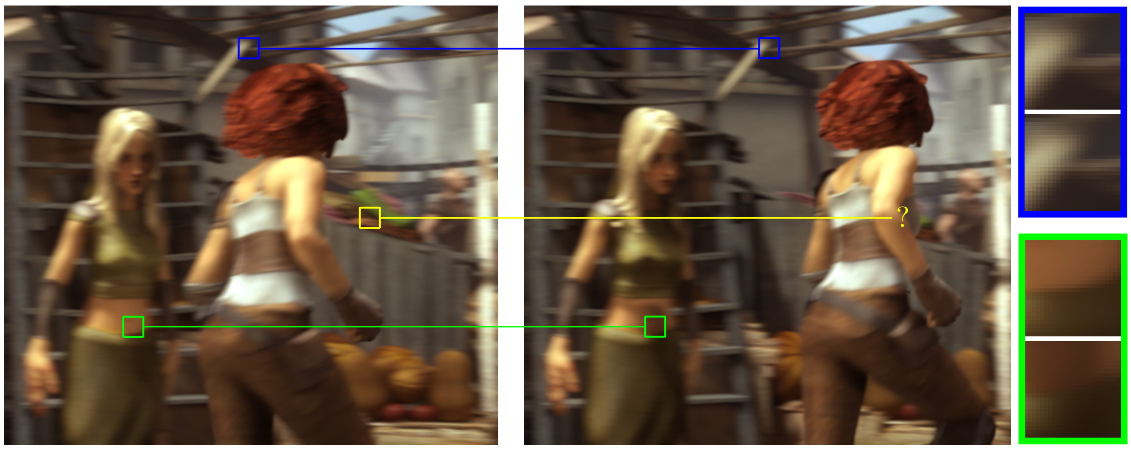


Figure 1.1: Corresponding patches in images. The blue patches look similar in both images, whereas the green patches change appearance. The yellow patch in the left image is occluded in the right image.

In order to see how these difficulties affect the mathematical properties of the correspondence problem, let us introduce a formal notation. We consider a domain $\Omega = \{1, \dots, W\} \times \{1, \dots, H\}$ of width W and height H on which we define two images $I^1, I^2 : \Omega \rightarrow \mathbb{R}^C$, where $C \in \{1, 3\}$ corresponds to grayscale and RGB-color images respectively. Pixel positions in the first and second image are denoted by $p, \hat{p} \in \Omega$ respectively and we use the notation $I(p)$ and I_p to access the value of the pixel at p. Moreover we define a function $\psi(\cdot\,; I) : \Omega \rightarrow \mathbb{R}^n$, that computes a *feature vector* (or *descriptor*) from a position p in an image I and we let $\Psi^{\{1,2\}} = \psi(\Omega, I^{\{1,2\}})$ be the features of the two images. Last, a distance function $d : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}_+$ computes the distance between two feature vectors in an n-dimensional space.

Then we can formulate the correspondence problem as

$$\min_{\hat{p}\in\Omega} d\left(\Psi^1(p),\ \Psi^2(\hat{p})\right), \tag{1.1}$$

which tries to find the position $\hat{p}$ in the second image, such that the distance between the feature vectors of p and $\hat{p}$ is minimized. Then we can hope that p and $\hat{p}$ are corresponding points. For example, consider the simple case that the descriptor is the pixel value, i.e. $\Psi(p) = I(p) \in \mathbb{R}^C$, and the distance function is given by the norm of the difference between the descriptors. Then the model is equivalent to minimizing the well-known Absolute Differences (AD) similarity measure, e.g. for $C = 1$ the problem is given by $\min_{\hat{p}\in\Omega} |I^1(p) - I^2(\hat{p})|$.

We can already see that the function ψ will play an important role. The goal is to make the feature vector as descriptive as possible. With patch-based similarity measures like Sum of Absolute Differences (SAD), it is clear that an increased window size results in higher quality matching, since comparing larger patches is more discriminative than comparing single pixel values. On the other hand, larger patches also result in larger deviations due to perspective, lighting changes, motion etc. More sophisticated descriptors like Normalized Cross-Correlation (NCC) [Lewis, 1995], Census [Zabih and Woodfill, 1994] or Scale-Invariant Feature Transform (SIFT) [Lowe, 2004] are to some extent invariant against illumination changes, rotation etc. and allow to deal with some of the problems caused by a). In fact, there is an ever-growing body of literature on how to compute good features for various problems in computer vision. This bears evidence that computing meaningful features is an active research topic in its own right.

A straightforward approach to solve the correspondence problem eq. (1.1) is by exhaustive search: Compare the descriptor of a pixel the first image to all descriptors in the second image and take the point whose descriptor gives the minimum distance as solution. If we want to establish correspondence for more than one pixel, eq. (1.1) lends itself to the notion of a matching cost. Given a set of points $P = \{p_k\},\ p_k \in \Omega,\ k = 1 \dots K$, we define the matching cost as

$$c(P, \hat{P}) = \sum_{k=1}^{K} d\left(\Psi^1(p_k),\ \Psi^2(\hat{p}_k)\right). \tag{1.2}$$

Then we can formulate the correspondence problem for multiple pixels conveniently as minimization of the matching cost for all points

$$\min_{\hat{P}} c(P, \hat{P}) \tag{1.3}$$

which is nothing more than repeatedly solving eq. (1.1). Note that every individual correspondence problem involves minimization over the complete image domain Ω, which quickly becomes very costly if the images are big and K is large. Therefore a reduced set of possibilities is often used in practice, which results in a substantial simplification. As an example, consider the classical computer vision problem of image stitching, where the search for corresponding points appears as a subproblem. A typical algorithm for image stitching consists of the following steps

- Obtain interesting points $P = \{p_k\},\ \hat{P} = \{\hat{p}_l\},\ k = 1 \dots K,\ l = 1 \dots L$ in the first and second image respectively, e.g. by running a keypoint detection algorithm. In the general case we have $K \neq L$, different numbers of keypoints in the first and second image.
- Establish correspondences by solving $\min_{\hat{p}_l \in \hat{P}} c(p_k, \hat{p}_l)$ for $k = 1 \dots K$. Here we have to minimize over the reduced set of points $\hat{P}$.
- Use the correspondences to compute a homography via some robust estimation procedure, e.g. Random Sample Consensus (RANSAC).
- Warp the images into the final panorama.

Running a keypoint detector on the image is a preprocessing step of the correspondence problem: It reduces the number of points for which we have to find correspondences to K, and the number of potential matches for each p_k to L.

1.2 Dense Image Matching

The preselection step via a keypoint detector will usually concentrate on *interesting* image points, that is, on corners or strong texture or points that exhibit some other salient property. The dense image matching problem is obtained by taking eq. (1.3) to the extreme: We skip the preprocessing step and wish to match all points in the image, without any constraint on the number of potential matches. This creates a new difficulty as illustrated in fig. 1.2.

c) Pixels from homogeneous image regions might look alike, hence different points will produce the same descriptor. This makes the matching ambiguous.

We conclude that we have neither stability (see a)) nor existence (see b)) nor uniqueness (see c)) of the solution. This makes the dense matching problem ill-posed, which means that in general it is not possible to compute a direct solution.

The standard approach when facing ill-posed problems is to pose an optimization problem via the following energy minimization formulation

$$\min_u \{E(u) = D(u) + R(u)\}. \tag{1.4}$$

Here the unknown u is a parameterization of correspondences that might take different forms depending on the application. For example, in stereo matching the images are preprocessed such that the epipolar lines are horizontal and parallel. Then correspondence

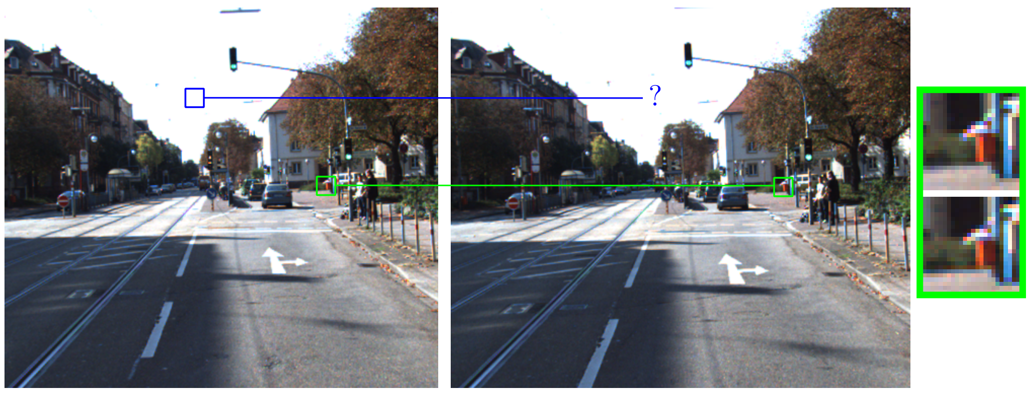


Figure 1.2: Corners and high-contrast texture help to make the correspondence unique (green patches). In homogeneous regions, the problem is ambiguous (blue patch).

can be parameterized by a scalar called disparity: $u_p \in \mathbb{R}, u_p = \hat{p}^{(1)} - p^{(1)}$, where $p^{(1)}$ denotes the first component (x-coordinate) of point p. In a more general case the parameterization could be a flow field that describes the 2D-displacement for every pixel, i.e. $u_p \in \mathbb{R}^2,\ u_p = \hat{p} - p$.

The term $R(u)$ is called *regularization term.* It originates from Bayesian Maximum a Posteriori (MAP) estimation, where it corresponds to the image prior model. We have seen that the dense matching problem is ill-posed, which means that there are infinitely many possible solutions. The purpose of the regularizer is to pick one particular solution among the infinitely many based on some prior assumption. For example, one widely used regularizer is the smoothness assumption, which states that neighboring pixels are likely to have a similar solution. This allows to compute a meaningful solution in regions where the dataterm is uninformative, see fig. 1.2. Modeling the regularizer is therefore of critical importance. It should be general enough for a wide range of input data, yet problem-specific in modeling desirable properties of the solution. This raises the question how to find a good tradeoff between these conflicting requirements. Chapter 3 will be devoted to this question in the context of stereo matching and reconstruction of grayscale images from event cameras.

The term $D(u)$ is called *data term*, it measures how well the solution u fits the underlying data, i.e. the images. It is given by the matching cost eq. (1.2), which in turn is based on the concept of a feature vector. In chapter 4 we will address the question how to find good features by means of deep learning with a Convolutional Neural Network (CNN).

1.3 Computing Solutions of the Dense Matching Problem

There are two fundamentally different approaches to solving the energy minimization eq. (1.4): Discrete and continuous optimization. Both have distinct advantages and disadvantages, stemming from the mathematical principles used to compute a solution, namely combinatorial optimization and calculus of variations. In the following we give a short overview of these two approaches.

1.3.1 Discrete Optimization

In the discrete setting, images are modeled as a graph, consisting of a vertex set $\mathcal{V} = \Omega$ corresponding to pixels, and an edge set $\mathcal{E} = \mathcal{V} \times \mathcal{V}$ corresponding to connections between pixels. The solution is given as the set $u = \{u_p\},\ p \in \mathcal{V}$, where every u_p takes (discrete) values out of some label space $\mathcal{L}$. This space is constructed depending on the problem at hand, e.g. in stereo matching it could be a discretization of the disparity range. The data term of the energy eq. (1.4) is given by the sum of the matching cost eq. (1.2) over all pixels, $D(u) = \sum_{p \in \mathcal{V}} c(p, \hat{p})$. The regularizer is defined as $R(u) = \sum_{kl \in \mathcal{E}} r(u_k, u_l)$, where the penalty function $r(\cdot, \cdot)$ is used to impose desirable properties on the solution.

In practice, the regularizer is computed on a n-connected local neighborhood, usually with $n \in \{4, 8, 16\}$, instead of exhausting the full space of all possible connections for every pixel. In a Bayesian framework, this means the probability distribution factorizes over the graph[2] and such models are are called *graphical models*. The two dominant approaches for solving graphical models in computer vision are MINCUT/MAXFLOW and message passing methods.

The MINCUT/MAXFLOW algorithm was first used in computer vision in the context of binary image restoration [Greig et al., 1989]. As the name suggests, the MINCUT algorithm computes a minimum cut in a graph $(\mathcal{V}, \mathcal{E})$ with vertices $\mathcal{V}$ and edges $\mathcal{E}$. The graph is endowed with a source vertex s, a sink vertex t and edge capacities $w : \mathcal{E} \to \mathbb{R}_+$. A minimum cut is a partition $\mathcal{V} = (S, T \setminus S)$ that separates the source vertex from the sink vertex and minimizes the sum of all boundary edge capacities, i.e. edges that start in S and end in T. By the famous max-flow min-cut theorem [Elias et al., 1956, Ford and Fulkerson, 1956], the value of the minimum cut is equal to the value of the maximum flow between the source and sink vertex. In fact, from an optimization perspective the maximum flow is the dual of the minimum cut. The limitation of the MINCUT algorithm to binary labelings was overcome by the works of [Boykov et al., 1998, Ishikawa and Geiger, 1998], who showed that the method could also be used to optimize a fairly general multilabel energy. This sparked a tremendous interest across all fields of computer vision, and to date graph cuts are among the most popular methods for discrete energy minimization due to their efficiency. Algorithmic advances include better runtime bounds [Boykov and Kolmogorov, 2004], move-making strategies [Lempitsky et al., 2010, Boykov et al., 2001] and even parallel multicore implementations [Delong and Boykov, 2008].

Message passing methods maximize the concave dual of the Linear Programming (LP) relaxation of the energy. Widely used algorithms are the Tree-Reweighted Max-Product Message Passing (TRW) [Wainwright et al., 2005] and its improved successor Sequential Tree-Reweighted Message Passing (TRW-S) [Kolmogorov, 2006]. Whereas the former does

[2] given its neighborhood, a variable is conditionally independent of the rest of the graph.

not converge monotonically, the latter does, but since both belong to the family of block-coordinate-ascent algorithms, they can get stuck in suboptimal points. In contrast to MINCUT/MAXFLOW, for which there is strong duality, in general there is no guarantee that, given a dual solution, a primal solution satisfying complementary slackness exists. However, if the energy is submodular, both MINCUT and message passing algorithms converge to the optimal solution.

Discrete algorithms are more challenging to parallelize than continuous methods. Even though block-coordinate-ascent algorithms like *TRW-S* can compute message updates parallel, it has been found that sequential updates often perform better in practice [Kappes et al., 2015]. There also exist methods that compute an energy dependent schedule of updates which results in faster convergence and better results [Tarlow et al., 2011], albeit complicating parallel implementations. On a higher level, the basic paradigm of cyclic maximization in each coordinate is not well suited for massive parallelism. However, recently there have been advances in this regard, with FPGA implementations of *TRW-S* [Choi and Rutenbar, 2012, Hurkat et al., 2015] and new massively parallel dual solvers that are implemented on the Graphics Processing Unit (GPU) [Shekhovtsov et al., 2016].

We point out that in the discrete setting, the data cost is precomputed by *sampling* the cost function at positions $k \in \Omega$. The full information is available to the optimization algorithm, and in some lucky cases it can even find a globally optimal solution. Because the data cost is precomputed for all possible positions, the matching cost function itself can be arbitrarily complex: complicated non-convex functions like *NCC* or *SIFT* are often used due to their good matching performance. The regularizer is usually restricted to first-order differences, since general higher order regularization is significantly more difficult to optimize and algorithms converge very slowly [Fix et al., 2014, Arora et al., 2012]. On the upside, robust, e.g. truncated and other non-convex regularization functions are easily dealt with by discrete optimization algorithms, as long as they are first-order.

1.3.2 Continuous Optimization

In the continuous setting images are modeled as functions $I : \Omega \to \mathbb{R}^d$, with $\Omega \subset \mathbb{R}^2$. In this case, the solution u itself is a function. The sum from the discrete setting is replaced by an integral and we obtain the continuous dataterm $D(u) = \int_\Omega c(u)\,\mathrm{d}x$. Likewise, the regularizer is given by $R(u) = \int_\Omega r(Ku)\,\mathrm{d}x$, where K is an *analysis* operator and $r(\cdot)$ is a (robust) penalty function. If we choose for example $K = \nabla$ and the absolute value function $r(\cdot) = |\cdot|$, we obtain the famous Total Variation (TV) regularizer $R(u) = TV(u) = \int_\Omega |\nabla u|\,\mathrm{d}x$, which enforces sparsity of gradients, i.e. smoothness, in the solution. In contrast to the discrete setting, where the pixel neighborhood played an integral role in defining the graph factorization and has strong algorithmic implications, in the continuous we do not need to worry about the neighborhood.

A large body of literature exists on the topic of minimization of convex functions, ranging from standard gradient descent to more complicated higher order methods with elaborate linesearch schemes [Bertsekas, 1999, Boyd and Vandenberghe, 2004, Nesterov, 2004]. However, since problems in computer vision are usually large-scale with millions of unknowns, and often non-smooth as well, most practically relevant algorithms are based on first-order schemes. These methods are easy to implement, since in each iteration they only need to evaluate the gradient. Moreover, many computer vision problems exhibit a natural grid-like structure which makes parallel implementations on *GPUs* straightforward.

Since edges are a fundamentally important image property, we wish for methods that are able to represent steps and sharp jumps. This means that many useful continuous models are inherently non-smooth, i.e. not continuously differentiable. To solve such large non-smooth problems, *proximal splitting* methods have been proven to be very effective. There exist many variants and extensions of this approach, all of which can be seen as instances of the basic Proximal Point Algorithm (PPA) [Martinet, 1970]. An extensive recent overview can be found in [Chambolle and Pock, 2016].

We note that even though most algorithms from convex optimization theory can also be applied in the non-convex case, we usually want our models to be convex. In the convex case any local minimum is also a global minimum, whereas in the non-convex case one can in general only compute critical points, e.g. local minima or saddle points. Globally optimal solutions are important, since they make interpretations of the models much easier. Hence, one important difference to the discrete setting is that in the continuous we usually restrict the data term as well as the regularization term to be convex functions.

The regularization function is typically convex by construction. In contrast to the discrete setting, where the regularizer was restricted to first-order differences, in the continuous we can also have higher order regularization: Total Generalized Variation (TGV) [Bredies et al., 2010] is a convex extension of TV. TGV allows to overcome the so-called staircaising effect of TV regularization. Staircaising occurs because the space of minimizers of TV is spanned by piecewise constant basis functions. TGV of order n enriches the space of basis functions with polynomials of up to order $n-1$, i.e. second-order TGV allows for piecewise affine solutions.

Turning to the matching cost, the data term usually involves the image function and is thus inherently non-convex. A widely used approach is to convexify the data term with a first-order Taylor expansion, i.e. linearization. Because the linearization becomes invalid far away from the linearization point, a coarse-to-fine warping strategy is used in order to deal with large displacements [Brox et al., 2004]. The idea is to approximate the original non-convex problem through a series of convex models with linearized data terms. However it is known that this approach suffers from loss of fine details, as small-scale structures that are not visible at coarse levels cannot be recovered later on. To overcome this drawback, methods have been developed that under certain circumstances are able to compute a globally optimal solution even with non-convex data term. [Pock et al., 2008] showed that if the solution space exhibits some natural ordering and the regularizer is the TV, it is possible to lift the problem to a higher-dimensional space where it becomes convex and can be solved globally. This idea has been extended to general convex functions of ∇u, e.g. [Pock et al., 2010, Ranftl et al., 2013].

Let us also mention that while first-order optimization algorithms are the tool of choice for solving large-scale problems in computer vision, straightforward implementations often suffer from slow convergence. A long line of research is devoted to acceleration tricks that can significantly improve convergence rates. Many of these tricks are based on extrapolation of gradients or "momentum", they are very easy to implement and typically do not change the computational complexity of the algorithm significantly. We mention the accelerated gradient descent method for smooth problems [Nesterov, 1983], the Fast Iterative Shrinkage Thresholding Algorithm (FISTA) algorithm by [Beck and Teboulle, 2009] for composite problems consisting of a smooth and a non-smooth function and the primal-dual algorithm of [Chambolle and Pock, 2011] for completely non-smooth problems.

In summary, continuous and discrete approaches have quite different properties due to the underlying mathematical principles as shown in table 1.1.

	Discrete	Continuous
Paradigm	combinatorial optimization	calculus of variations
Model domain	$\Omega = \{1 \dots H\} \times \{1 \dots W\}$	$\Omega \subset \mathbb{R}^2$
Solution	set $u = \{u_p \in \mathcal{L}\},\ p \in \Omega$	function $u \in H^{1,1}(\Omega)$
Data term	arbitrary (sampled)	convex (linearized)
Regularization term	first-order, non-convex	higher-order, convex
Large displacements	easy	hard
Parallelization	hard	easy
Memory consumption	high	low

Table 1.1: Discrete vs. continuous optimization

1.4 Contributions and Outline

The dense matching problem is ill-posed and is solved by means of an optimization problem eq. (1.4). The energy to be minimized consists of the regularization term and the data term, and the properties of each as well as the interplay between the two terms determines the quality of the solution. In this thesis we make contributions to both terms:

- **Regularization** We consider a geometric approach and develop a novel regularizer based on the inner geometry of the solution manifold. This results in a physically meaningful geometric prior, which turns out to be useful in problems with a strong geometric background. Whereas physically meaningful priors are relatively easy to compute given an explicit representation of the surface, the resulting algorithms are often slow and require complicated and resource-hungry data structures and considerable overhead in order to maintain the representation of the surface. In contrast, we propose to formulate the regularizer on the image plane using techniques from differential geometry. We build on the fact that surfaces in 3D can be parameterized by a depthmap. Such parameterization corresponds to the notion of *charts* from manifold theory. We show that the metric tensor allows to pull back problems defined on the surface to the local coordinate system of the image plane. This enables efficient and highly parallel implementations on *GPUs*, without the need to maintain an explicit representation of the surface.

- **Data term** We tackle the problem of learning descriptors for dense optical flow with a *CNN*. As pointed out in section 1.1, the descriptors play a crucial role in the matching problem. Learned descriptors generalize over hand-crafted ones by providing problem-specific features. To avoid a patch-sampling step, we opt for a general one-vs-all learning approach, which previously showed good results in the context of stereo matching [Luo et al., 2016]. However, implementing such approach for optical flow turns out to be difficult, because one-vs-all learning needs the full cost function in memory. The size of the cost function grows quadratically with the search range, which is feasible in the context of stereo matching, but intractable for optical flow. Therefore we propose a dimensionality reduction of the optical flow cost function. This drastically reduces memory requirements and makes end-to-end learning on high resolution images tractable. To reduce the computational complexity, we consider binary feature vectors and introduce a new learning scheme which improves upon existing approaches for learning binary *CNNs*.

Outline Chapter 2 of this thesis gives an overview of the notation and the mathematical foundations needed in the rest of the work. We give a short introduction to convex optimization in section 2.1, followed by the basics of discrete optimization, section 2.2. Section 2.3 is devoted to differential geometry, needed for the metric regularization of surfaces.

In chapter 3 we present two applications of our regularizer: First, we consider the dense stereo matching problem in section 3.1. The goal is to compute a 3D surface from a pair of images. In this context, popular image-based regularizers like *TV* or *TGV* are agnostic of the fact that the object of interest is a 3D surface. Inspired by the research on minimal surfaces, we propose to use the area of the surface as regularizer. To that end, we derive the surface area form under orthographic and perspective projection and show how to compute it in local coordinates, i.e. on the image plane. Our regularizer, while formulated on the image plane just like *TV*, respects the geometry of the surface. We show that this enables higher-quality 3D reconstructions. Second, we apply the metric regularizer to construct grayscale images from event cameras in section 3.2. Event cameras or neuromorphic cameras are different from standard frame-based cameras, they operate asynchronously on the pixel level. Each pixel measures the incoming light and fires an event in case the absolute change in intensity is above a threshold. Thus, in a given time interval one gets only a sparse set of events carrying the binary information {lighter (+1), darker (-1)} instead of a full frame. Among the advantages of the event camera are the high time resolution and the excellent dynamic range. However, visualizing the stream of events as sparse black and white pixels, corresponding to the events -1 and +1, is not very informative. There is a need for reconstructing grayscale images from the event stream for visualization and verification purposes. We apply the metric regularizer in the variational energy used for the reconstruction by incorporating additional information from the so-called manifold of active events. This allows to formulate the variational model on the surface given by the time history of the events and results in higher contrast and sharper edges of the reconstructed images.

Chapter 4 is devoted to learning descriptors for optical flow with a *CNN*. We show how one-vs-all learning is enabled by a dimensionality reduction via partial optimization of the local optical flow matching cost. After verifying that the learned descriptors give reasonable results using a simple Winner-Takes-All (WTA) strategy, we consider a discrete energy minimization model that imposes robust regularization via a Conditional Random Field (CRF). Here we face a similar problem as in the learning phase: Computing solutions of the *CRF* needs the full cost function in memory. We adapt the dimensionality reduction technique from *CNN* learning for *CRF* inference, and show how to efficiently minimize the model with a highly parallel dual solver. The approach is based on a graph decomposition, where the two components of the flow vector are modeled by two three-dimensional stereo-like problems coupled through the four dimensional cost function. Our approach enables efficient *CRF* inference on high resolution images with the full quadratic label space at linear memory complexity.

2 Mathematical Foundations

Contents

The purpose of this chapter is to introduce the notation and mathematical topics needed in the rest of this work. We start with convex analysis and convex optimization, which form the underlying theory of the variational models used to compute a solution of the dense matching problem. Next, we give an overview of discrete optimization algorithms. Finally we develop basic concepts of differential geometry, which are needed for the metric regularization of surfaces in chapter 3.

In the sections about convex and discrete optimization we use the space $\mathbb{R}^n$ with its vector space structure, i.e. component-wise addition and scalar multiplication of points. In order to be consistent with the usual notation in differential geometry, we denote $x = (x^1, \ldots, x^n) \in \mathbb{R}^n$, i.e. superscripts denote components, not exponentiation. We will also use the standard Euclidean inner product $\langle \cdot, \cdot \rangle : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} : \langle x, y \rangle \mapsto \sum_{i=1}^n x^i y^i$ on $\mathbb{R}^n$.

2.1 Convex Optimization

The material in this section is based on the textbooks [Rockafellar, 1970, Nesterov, 2004, Boyd and Vandenberghe, 2004].

2.1.1 Norms

Definition 2.1. Let V be a real vector space. A *norm* on V is a function, written $\| \cdot \| : V \to \mathbb{R}$, that satisfies $\forall v, w \in V$

1. *Positivity:* $\|v\| \geq 0$.
2. *Definiteness:* $\|v\| = 0$ iff $v = 0$.
3. *Homogeneity:* $\|\lambda v\| = |\lambda|\|v\|$, with $\lambda \in \mathbb{R}$.
4. *Triangle inequality:* $\|v + w\| \leq \|v\| + \|w\|$.

If condition 2 (definiteness) is violated, the function is called *semi-norm.* A vector space with a norm is called *normed space.* The most widely used norm is arguably the *Euclidean norm* or *standard norm* on $\mathbb{R}^n$, defined as

$$\|v\| = \sqrt{\sum_{i=1}^{n} (v^i)^2}\,. \tag{2.1}$$

The Euclidean norm is related to the Euclidean inner product by $\|v\| = \sqrt{\langle v, v\rangle}$. We say that the Euclidean norm is *induced* by the Euclidean inner product. Furthermore, the Euclidean norm is a special case of the more general ℓ_p-norm given by

$$\|v\|_p = \left(\sum_{i=1}^{n} |v^i|^p \right)^{1/p}, \quad p \in [1, \infty), \tag{2.2}$$

where the Euclidean norm is obtained for $p = 2$. The Euclidean inner product and Euclidean norm have a strong connection to geometry, which will be discussed in much more detail in section 2.3, where we give an introduction to differential geometry. Here we just note that the Euclidean norm coincides with our usual notion of length.

As customary, we may omit explicit specification of $p = 2$ for the standard Euclidean norm and write $\|\cdot\| = \|\cdot\|_2$.

The special case $p = \infty$ yields the so-called *infinity norm*, defined by

$$\|v\|_\infty = \max_{i=1,\dots,n} \{|v^i|\}, \tag{2.3}$$

i.e. the infinity norm is the maximal absolute component of v.

Definition 2.2 (Operator norm)**.** Let $T : V \to W$ be a linear mapping, called *operator*, between real vector spaces V, W. Every linear operator can be represented as a matrix $T \in \mathbb{R}^{(\dim W)\times(\dim V)}$. The operator norm is defined as

$$\|T\| = \inf\{M \in \mathbb{R} : \|Tv\| \leq M\|v\| \ \forall v \in V\}. \tag{2.4}$$

The operator norm is a measure how much the norm of a vector changes under the mapping T. Equivalently, the operator norm can be given by $\|T\| = \sup_{\|v\|\leq 1} \{\|Tv\|\} = \sup_{\|v\|=1}\{\|Tv\|\}$. Note that the operator norm depends on the choice of norm in the two spaces V and W. For example, if we choose the Euclidean norm in both V and W, the operator norm is commonly denoted $\|\cdot\|_{2,2}$ and the value corresponds to the largest singular value of the matrix T.

Definition 2.3 (Dual norm)**.** Let $\|\cdot\|$ be a norm on a real vector space V and $v, w \in V$. The dual norm $\|\cdot\|_*$ is defined as

$$\|v\|_* = \sup_{\|w\|\leq 1} \{\langle v, w\rangle\} = \sup_{\|w\|=1} \{\langle v, w\rangle\} \tag{2.5}$$

The resemblance of the dual norm with the definition of the operator norm is not a coincidence. If v is represented as a column vector, we can interpret the row vector v^T as an element of the dual space V^*, i.e. a linear functional. Elements of the dual space are also called covectors, and a covector field is known as *differential one-form* in the context of differential geometry, see section 2.3.5.1. The action of a covector on a vector is given by the inner product, i.e. a covector "eats" a vector and returns a scalar. Thus we can interpret the dual norm as the operator norm of the linear functional v^T.

The ℓ_p-norm $\|\cdot\|_p$ and its dual norm $\|\cdot\|_q$ are related by $1/p + 1/q = 1$, e.g. the 2-norm is self-dual and the dual norm of $\|\cdot\|_1$ is $\|\cdot\|_\infty$.

2.1.2 Convex Sets

Definition 2.4. A subset $C \subseteq \mathbb{R}^n$ is *convex*, if for all $x, y \in C$ and $\theta \in [0, 1]$

$$\theta x + (1-\theta)y \in C.$$

Intuitively this definition means that the straight line between any two points x, y in the set is completely in the set as well, see fig. 2.1. By convention, the empty set $\{\emptyset\}$ and the whole space $\mathbb{R}^n$ are convex sets.

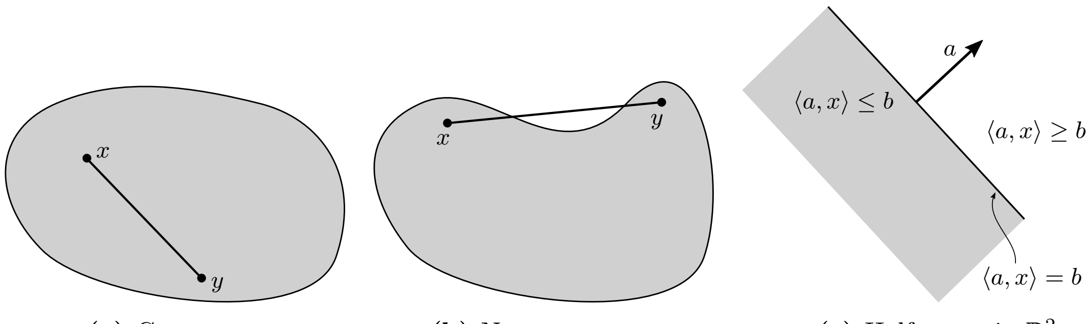

x
y
x
y
a
$\langle a, x\rangle \leq b$
$\langle a, x\rangle \geq b$
$\langle a, x\rangle = b$
(a) Convex set **(b)** Non-convex set **(c)** Halfspaces in $\mathbb{R}^2$.

Figure 2.1: Sets in the two-dimensional plane.

An important concept is the *hyperplane*, defined as the set $\{x \in \mathbb{R}^n \mid \langle a, x\rangle = b\}$ with $a \neq 0 \in \mathbb{R}^n$ and $b \in \mathbb{R}$. A hyperplane can be visualized as the plane with normal vector a

and distance from the origin $\frac{b}{\|a\|}$. It divides the space $\mathbb{R}^n$ into two *halfspaces* of the form $\{x \in \mathbb{R}^n \mid \langle a, x\rangle \gtreqless b\}$, both of which are convex sets, see fig. 2.1(c). The following lemma establishes a connection between a convex set and the halfspaces which contain it.

Lemma 2.1. *Let S be a convex set and denote by $\mathcal{H}$ the set of all halfspaces which contain S. Then $S = \bigcap_{H\in\mathcal{H}} H$.*

Proof. By definition the halfspaces contain S, so clearly $S \subseteq \bigcap_{H\in\mathcal{H}} H$. To prove that $S \supseteq \bigcap_{H\in\mathcal{H}} H$, we need to show that if $x \in \bigcap_{H\in\mathcal{H}} H$ it follows that $x \in S$, or, equivalently, if $x \notin S$ it follows that $x \notin \bigcap_{H\in\mathcal{H}}$. By the separating hyperplane theorem, if $x \notin S$ there is a hyperplane separating x from S. This hyperplane defines a halfspace H containing S, and since $x \notin S$ it follows $x \notin H$ which implies $x \notin \bigcap_{H\in\mathcal{H}} H$. □

If we have a number of points $\{x_1, \ldots, x_K\}$ and positive scalars $\theta_1, \ldots, \theta_K$ with $\sum_k \theta_k = 1$, we call $x = \sum_k \theta_k x_k$ a *convex combination* of points. Any convex combination of points taken from a convex set is again in the set. Conversely, a set is convex if and only if it contains all convex combinations of its points.

In dealing with convex sets, it is important to know which operations preserve convexity. We have for convex sets C_1, C_2 that

- The intersection $C_1 \cap C_2$ is a convex set.
- The vector sum $C_1 + C_2 = \{x + y : x \in C_1, y \in C_2\}$ is a convex set.
- The image $f(C_1) = \{f(x) \mid x \in C_1\}$ under an affine function $f(x) = Ax + b$ with $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$ is a convex set.

2.1.3 Convex Functions

Definition 2.5. Let $f : S \to \mathbb{R}$ be a function with domain $S \subseteq \mathbb{R}^n$. The set

$$\operatorname{epi} f = \{(x, t) \in \mathbb{R}^n \times \mathbb{R} \mid x \in S, t \geq f(x)\}$$

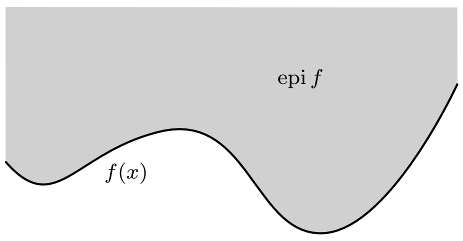
is called the *epigraph* of f.

The prefix “epi” from ancient greek means “on top of”, hence the epigraph is the set of all points above the graph of f, see fig. 2.2(a). Note that the epigraph of a function with domain $S \subseteq \mathbb{R}^n$ is a subset of $\mathbb{R}^{n+1}$.

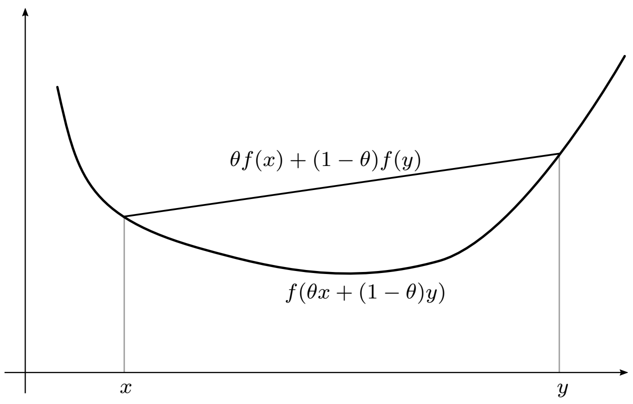
With the help of the epigraph, we now define the central object of convex analysis.

Definition 2.6 (Convex function)**.** Let $S \subseteq \mathbb{R}^n$. A function $f : S \to \mathbb{R}$ is *convex* if its epigraph is a convex set. We call f *concave* if $(-f)$ is convex.

From the convexity-preserving operations on sets in the previous section we get an important relation between the convexity of a function and its domain.



(a) The epigraph of f.



(b) The straight line between the two points $f(x), f(y)$ is always above the graph of the convex function f.

Figure 2.2: Epigraph (a) and convex function (b).

Lemma 2.2. *The domain $S \subseteq \mathbb{R}^n$ of a convex function $f : S \to \mathbb{R}$ is a convex set.*

Proof. Since f is convex, its epigraph is a convex set. Define $L : S \times \mathbb{R} \to S : (x,t) \mapsto x$ as the mapping that projects the epigraph of f onto the domain of f. Clearly L is linear, and since convexity of a set is preserved under linear mappings, it follows that the domain S is a convex set. □

In particular, this means that in order for a function to be convex, its domain must be a convex set. Sometimes this condition is included into the definition of a convex function. Besides the basic definition 2.6 there are a number of other characterizations of convex functions, many of which have a strong geometric interpretation. In the following we will list a few of them.

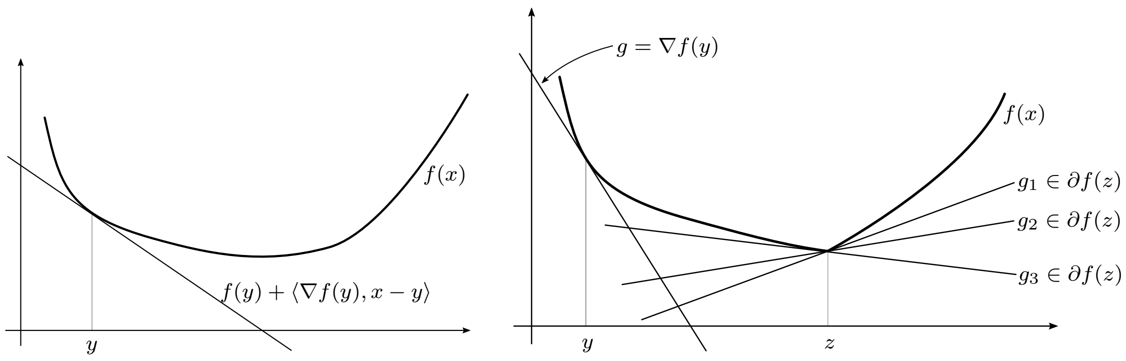
Jensen's Inequality We start with the fact that convex functions enjoy an important interpolation property known as Jensen's inequality. Namely, if f is a convex function with (convex) domain $S \subseteq \mathbb{R}^n$, it holds that

$$\forall x, y \in S,\ \theta \in [0,1] : \quad f(\theta x + (1-\theta)y) \le \theta f(x) + (1-\theta) f(y). \tag{2.6}$$

Geometrically this means that the straight line between any two points $f(x)$ and $f(y)$ is always above the graph of f, see fig. 2.2(b). Due to this intuitive interpretation, eq. (2.6) is often used as an alternative definition of convexity.

First-Order Condition If a convex function $f : S \to \mathbb{R},\ S \subseteq \mathbb{R}^n$ is differentiable, it can be characterized as follows

$$\forall x, y \in S : \quad f(x) \ge f(y) + \langle \nabla f(y), x - y \rangle\,. \tag{2.7}$$



(a) The graph of a convex function is always above its linearization.

(b) The subdifferential is the set of all subgradients.

Figure 2.3: First-order convexity condition and subdifferential.

This means that the graph of a convex function is always above the linear approximation at any point y. In other words, f can be globally underestimated by its first-order approximation, see fig. 2.3(a). Moreover, if we switch the roles of the points x and y in eq. (2.7), we obtain through a short computation that

$$\forall x, y \in S : \quad \langle \nabla f(x) - \nabla f(y), x - y \rangle \geq 0, \tag{2.8}$$

which shows that the gradient of a convex function is monotone. The relations eqs. (2.7) and (2.8) play an important role in the optimization of convex functions, because they allow to obtain global information from the local gradient.

We can generalize eq. (2.7) for convex functions which are not continuously differentiable. In particular, we call any vector g that fulfills

$$\forall x \in S : \quad f(x) \geq f(z) + \langle g, x - z \rangle \tag{2.9}$$

at a point $z \in S$ a *subgradient* of f at z, denoted $g \in \partial f(z)$. If f is differentiable at z, then there is exactly one g that fulfills eq. (2.9), namely the vector $g = \nabla f(z)$. In this case eq. (2.9) is equivalent to eq. (2.7). If f is not differentiable at z there are multiple different subgradients that fulfill eq. (2.9), see fig. 2.3(b). The set of all subgradients is called the *subdifferential*. The subdifferential of a convex function is non-empty everywhere, a fact which allows to characterize the optimality condition of a convex function.

Definition 2.7. A point $x^* \in S \subseteq \mathbb{R}^n$ is a minimum of the convex function $f : S \to \mathbb{R}$ if and only if $0 \in \partial f(x^*)$. Therefore, if x^* is a minimum we have by eq. (2.9) that

$$\forall x \in S : \quad f(x) \geq f(x^*) + \langle \partial f(x^*), x - x^* \rangle = f(x^*)$$

which means that x^* is a global minimum of f. It follows that every (local) minimum of a convex function is also a global minimum.

Second-Order Condition If a convex function $f : S \to \mathbb{R}$, $S \subseteq \mathbb{R}^n$ is twice differentiable, it can be characterized by

$$\forall x \in S : \quad \nabla^2 f(x) \succcurlyeq 0,$$

the Hessian matrix is positive semidefinite everywhere.

To check convexity of a function f, in practice it is often useful to construct f from functions known to be convex and operations that preserve convexity. Some of these operations are

- The non-negative weighted sum of convex functions is convex.
- A convex function f under an affine mapping is convex, $g(x) = f(Ax + b)$, $A \in \mathbb{R}^{m\times n}, b \in \mathbb{R}^m$.
- The point-wise maximum of convex functions f_1, f_2 is convex, $g(x) = \max\{f_1(x), f_2(x)\}$.
- Norms are convex.

2.1.4 The Convex Conjugate

The convex conjugate is one of the most fundamental geometric relations in convex analysis. It is based on an alternative characterization of a convex set, namely as an intersection of all halfspaces containing it, see lemma 2.1. This alternative characterization is an instance of duality, which we already encountered in section 2.1.1 where we defined the dual norm. Applying the concept of duality to the epigraph of a convex function (which is a convex set) yields the convex conjugate. Recall that a halfspace is generated by a hyperplane of the form $\langle a, x\rangle - b = 0$, see section 2.1.2. Duality suggests that we are interested in characterizing the epigraph of a convex function f as an intersection of halfspaces.

Lemma 2.3. *A convex function $f : S \to \mathbb{R}$, $S \subseteq \mathbb{R}^n$ is the point-wise supremum of all affine functions minorizing it.*

Proof. Since f is convex, $\operatorname{epi} f$ is a convex set. By lemma 2.1, $\operatorname{epi} f$ can be represented as the intersection of halfspaces which contain it. These halfspaces are given as a parameterized family of the form $\langle a, x\rangle - t \le b$ with parameter $t \in \mathbb{R}$ and $a \in \mathbb{R}^n, b \in \mathbb{R}$. Each of the halfspaces constrains $(x, t) \in \operatorname{epi} f$ to $\langle a, x\rangle - b \le t$. The value $f(x)$ is the smallest value of t such that $(x, t) \in \operatorname{epi} f$, hence $f(x)$ is the point-wise supremum of affine functions $\langle a, x\rangle - b$. The fact that the halfspaces contain $\operatorname{epi} f$ implies $\langle a, x\rangle - b \le f(x)$, which shows that the affine functions are minorants of f. □

Note that together with lemma 2.1 this lemma also shows that the point-wise supremum of affine functions is the intersection of their epigraphs. Moreover, it is not hard to see that in fact we do not need all minorants to characterize f. It suffices to take the point-wise supremum of all tight minorants. For any minorant that is not tight, we can find another one with a smaller value of t such that (x, t) is still in epi f.

From lemma 2.3, we are interested in affine functions $x \mapsto \langle a, x\rangle - b$ that minorize f

$$\forall x : \quad f(x) \geq \langle a, x\rangle - b. \tag{2.10}$$

Fixing a, we can find the special value b^* such that eq. (2.10) holds with equality for some x_0 – this is exactly the tight affine minorant or the *supporting hyperplane* of epi f at x_0, see fig. 2.4. We should mention here that a is an element of the dual space, i.e. a covector.

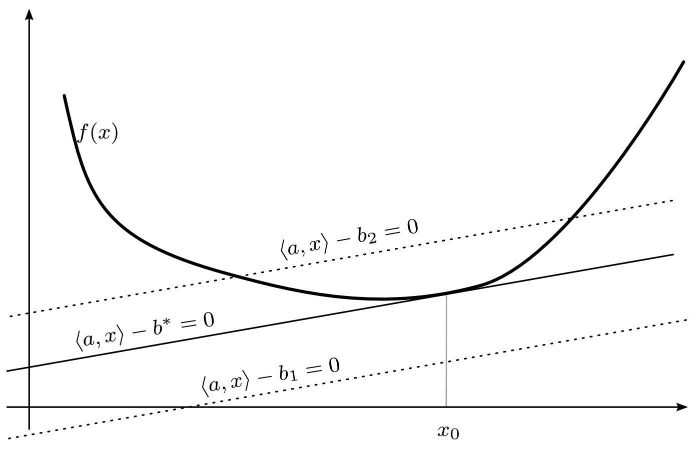

$f(x)$
$\langle a, x\rangle - b_2 = 0$
$\langle a, x\rangle - b^* = 0$
$\langle a, x\rangle - b_1 = 0$
x_0

Figure 2.4: Hyperplanes with fixed slope a and varying parameters b. The value b_1 yields a minorant of f, but the minorant is not tight. b^* gives a tight minorant, i.e. there is some x_0 for which eq. (2.10) holds with equality. The hyperplane with parameter b_2 does not minorize f.

Moreover we use the usual notation with angular brackets for the action of a covector on a vector, i.e. it should be understood that $\langle a, x\rangle = a(x)$. In geometric terms, a corresponds to the *slope* of the supporting hyperplane. Now we reorder terms in eq. (2.10) and obtain

$$\begin{aligned} & f(x) \geq \langle a, x\rangle - b && \forall x \\ \Leftrightarrow\ & b \geq \langle a, x\rangle - f(x) && \forall x \\ \Leftrightarrow\ & b \geq \sup_x \{\langle a, x\rangle - f(x)\}. \end{aligned} \tag{2.11}$$

The special value b^*, i.e. the parameter of the supporting hyperplane, is attained when equality holds

$$b^* = \sup_x \{\langle a, x\rangle - f(x)\}. \tag{2.12}$$

We can express the value of b^* in eq. (2.12) as a function of the dual variable, which yields

Definition 2.8 (Convex conjugate)**.** Let $S \subseteq \mathbb{R}^n$ with dual space $S^* = L(S, \mathbb{R})$ the space of all linear functionals on S. The convex conjugate $f^* : S^* \to \mathbb{R}$ of a function $f : S \to \mathbb{R}$

is defined as

$$f^*(y) = \sup_{x \in S}\{\langle x, y\rangle - f(x)\}.$$

The geometric interpretation of the convex conjugate is as follows: We feed a slope y to f^* and it returns the parameter b^*, i.e. such that the hyperplane with slope y minorizes f tightly, see fig. 2.4. The value of b^* corresponds to the intercept on the $(n+1)$-axis of the supporting hyperplane with slope y. Note that the convex conjugate is a supremum of affine functions, hence f^* is convex even if f is not. Inserting $a = y$ and $b^* = f^*(y)$ into eq. (2.11) and following the steps backwards, one immediately obtains the Fenchel-Young inequality $f(x) + f^*(y) \geq \langle x, y\rangle \quad \forall x \in S, y \in S^*$.

Applying definition 2.8 to the convex conjugate itself yields the biconjugate

$$f^{**}(x) := (f^*)^*(x) = \sup_{y \in S^*}\{\langle y, x\rangle - f^*(y)\}. \tag{2.13}$$

By the Fenchel-Young inequality, $f(x) \geq \langle y, x\rangle - f^*(y)$ for all $y \in S^*$, hence taking the supremum over y does not change the inequality and it follows that

$$f(x) \geq \sup_{y \in S^*}\{\langle y, x\rangle - f^*(y)\} = f^{**}(x), \tag{2.14}$$

the biconjugate is the convex envelope of f.

As a special and important case consider that f is convex. Fixing a slope y, by definition the convex conjugate is the value $f^*(y)$ such that $\langle x, y\rangle - f^*(y)$ is a tight minorant of f. Therefore the biconjugate eq. (2.13) is a supremum of tight minorants, and since f is convex, by lemma 2.3 this supremum is equal to f. We thus obtain the fundamental property that $f = f^{**}$, the biconjugate of a convex function is the function itself.

2.1.5 The Proximal Operator

One subproblem that often occurs in convex optimization is the projection onto a convex set. Assume we have a convex set $C \subseteq \mathbb{R}^n$ and a point $\tilde{x} \notin C$. The projection of $\tilde{x}$ onto C is given by

$$\operatorname{proj}_C(\tilde{x}) = \arg\min_{x \in C} \|x - \tilde{x}\|. \tag{2.15}$$

This is also called orthogonal projection, because under the Euclidean norm the shortest path from C to the point $\tilde{x}$ is the line orthogonal to the boundary of C, see fig. 2.5. Note that the norm is a convex function and we minimize over a convex set, hence eq. (2.15) is a constrained convex optimization problem which has a unqiue solution. We can alternatively transform eq. (2.15) to an unconstrained minimization problem using the indicator

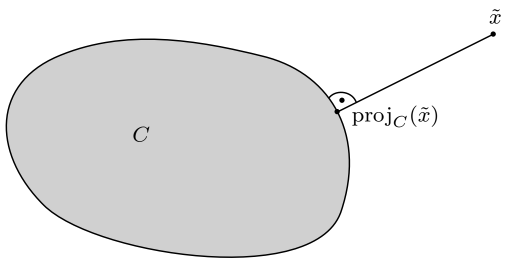


Figure 2.5: The orthogonal projection of a point $\tilde{x}$ onto the convex set C.

function of a set, which is defined as

$$I_C(x) = \begin{cases} 0 & \text{if } x \in C \\ \infty & \text{else} \end{cases} .$$

We get

$$\begin{aligned} \mathrm{proj}_C(\tilde{x}) &= \arg\min_x \{\|x - \tilde{x}\| + I_C(x)\} \\ &= \arg\min_x \left\{\tfrac{1}{2}\|x - \tilde{x}\|^2 + I_C(x)\right\}, \end{aligned} \tag{2.16}$$

since the minimizer does not change if we square the norm or multiply it by a constant.

The proximal operator is a generalization of eq. (2.16), where the indicator function is replaced by a general convex function.

Definition 2.9. The proximal operator with respect to a convex function f is defined as

$$\mathrm{prox}_f(\tilde{x}) = \arg\min_x \left\{\tfrac{1}{2}\|x - \tilde{x}\|^2 + f(x)\right\}. \tag{2.17}$$

The proximal operator is also often denoted as $(I + \partial f)^{-1}(\tilde{x})$, which can be seen by considering the optimality condition of eq. (2.17)

$$\begin{aligned} & 0 \in x - \tilde{x} + \partial f(x) \\ \Leftrightarrow & \tilde{x} \in x + \partial f(x) \\ \Leftrightarrow & \tilde{x} \in (I + \partial f)(x) \\ \Leftrightarrow & x = (I + \partial f)^{-1}(\tilde{x}) = \mathrm{prox}_f(\tilde{x}), \end{aligned}$$

where I denotes the identity operator. In the context of optimization algorithms, the notation $\mathrm{prox}_{\tau f}$ is frequently used, which simply means that the term $f(x)$ in eq. (2.17) is multiplied by a scalar τ.

We point out that the proximal operator does not assume differentiability of f. This will become important in the context of optimization of non-smooth functions.

2.1.6 Algorithms

In this section we give a few standard algorithms for optimizing a convex function. After presenting the basic gradient descent algorithm for smooth functions, we turn to proximal splitting methods for composite problems which consist of a smooth and a non-smooth part. Finally we present an efficient primal-dual algorithm for completely non-smooth problems.

The general form of a convex optimization problem is defined as

$$\min_{x \in C} f(x), \tag{2.18}$$

where $f : S \subseteq \mathbb{R}^n \to \mathbb{R}$ is a convex function and $C \subseteq \mathbb{R}^n$ is a convex set. Note that in addition to f being a convex function we also require that we optimize over a convex set. If $S \subset \mathbb{R}^n$ (i.e. S is a proper subset, different from $\mathbb{R}^n$ itself) then it is always possible to extend f to all of $\mathbb{R}^n$ by setting the function value to ∞ outside of S. In the rest of this section we will therefore drop the set S and assume that the objective function is defined on $\mathbb{R}^n$.

2.1.6.1 Gradient Descent

The most basic approach to optimizing a smooth convex function is simply taking steps along the negative gradient direction, i.e. in a direction where the function value decreases. This procedure is known as gradient descent, see algorithm 2.1. The advantage of the

Algorithm 2.1: Gradient descent

```
  Data: Convex differentiable function $f$
1 Initialization: Set $k = 0$, choose a starting point $x_0 \in \mathbb{R}^n$ and stepsizes $\alpha_k > 0$
2 while not converged do
3     $x_{k+1} = x_k - \alpha_k \nabla f(x_k)$
4     $k = k + 1$
5 end
```

gradient descent algorithm is that it is very simple. Of course the function needs to be differentiable in order to compute the gradient. If in addition $\nabla f(x)$ is Lipschitz continuous with parameter L, meaning L fulfills $\forall x, y \in \mathbb{R}^n : \quad \|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|$, then it can be proved that the gradient descent algorithm with constant stepsize $\alpha_k \in (0, 2/L)$ converges with rate $\mathcal{O}(1/k)$ to the optimal value.[1]

It is known that the lower complexity bound for any first-order method on smooth convex functions is $\mathcal{O}(1/k^2)$ [Nesterov, 2004]. This raises the question if algorithm 2.1 can be improved for faster convergence.

[1]Unless stated otherwise, we will always talk about convergence of the function value.

In a seminal work, [Nesterov, 1983] showed that one can achieve the optimal $\mathcal{O}(1/k^2)$ convergence rate with a simple extrapolation and a clever selection of the extrapolation parameters, see algorithm 2.2. The accelerated algorithm comes with basically unchanged

Algorithm 2.2: Nesterov accelerated gradient descent

```
  Data: Convex function $f$, with $L$-Lipschitz gradient
1 Initialization: Set $k = 0$, choose points $y_0 = x_{-1} \in \mathbb{R}^n$, $\alpha_0 = 1$
2 while not converged do
3     $x_k = y_k - \frac{1}{L}\nabla f(y_k)$
4     $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$
5     $y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(x_k - x_{k-1})$
6     $k = k + 1$
7 end
```

computational complexity, only memory complexity increases slightly because one needs to store the additional point y_k.

2.1.6.2 Proximal Methods

Proximal methods were invented out of the need to optimize non-differentiable functions, in which case gradient-based methods could not be applied. In the most basic form, one simply iterates

$$x_{k+1} = \operatorname{prox}_{\alpha f}(x_k),$$

which results in the Proximal Point Algorithm (PPA) [Martinet, 1970]. With appropriate stepsize, the *PPA* eventually converges to a fixed point of f. However, solving the proximal operator is usually as hard as optimizing the original function f, which means the *PPA* in this form it is a rather conceptual algorithm. We still mention it here, because many state of the art algorithms for non-smooth optimization of convex functions can be seen as an instance of the basic *PPA*.

Consider the class of optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \{f(x) = g(x) + h(x)\}, \tag{2.19}$$

where the objective function is given as the sum of a differentiable convex function $g(x)$ and a convex, possibly non-differentiable function $h(x)$. As it turns out, many practically relevant problems can be given in this form. Assume we have an estimate of a Lipschitz constant L of $\nabla g(x)$, then a simple algorithm that exploits the additional structure of the optimization problem eq. (2.19) is the proximal gradient method, see algorithm 2.3. It can be seen as a counterpart of the basic gradient descent algorithm 2.1. The proximal

Algorithm 2.3: Proximal gradient method

```
  Data: Convex function $g$ with $L$-Lipschitz gradient, convex function $h$
1 Initialization: Set $k = 0$, choose a starting point $x_0 \in \mathbb{R}^n$ and stepsizes $\alpha_k > 0$
2 while not converged do
3     $x_{k+1} = \mathrm{prox}_{\alpha_k h}\left(x_k - \alpha_k \nabla g(x_k)\right)$
4     $k = k + 1$
5 end
```

Algorithm 2.4: Fast Iterative Shrinkage Thresholding Algorithm (FISTA)

```
  Data: Convex function $g$ with $L$-Lipschitz gradient, convex function $h$
1 Initialization: Set $k = 0$, choose points $y_0 = x_{-1} \in \mathbb{R}^n$, $\alpha_0 = 1$
2 while not converged do
3     $x_k = \mathrm{prox}_{h/L}\left(y_k - \frac{1}{L}\nabla g(y_k)\right)$
4     $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$
5     $y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(x_k - x_{k-1})$
6     $k = k + 1$
7 end
```

gradient method works by taking gradient descent steps on the smooth part g, followed by a proximal step on the non-smooth part h. Such methods are known as *proximal splitting algorithms*. The proximal gradient method converges for $\alpha_k \in (0, 2/L)$ with rate $\mathcal{O}(1/k)$. It is interesting that even though the function $h(x)$ is not differentiable, the proximal gradient algorithm enjoys the same convergence rate as the gradient descent algorithm for smooth functions.

To improve upon the slow convergence, the proximal gradient method can be accelerated, see algorithm 2.4. This accelerated algorithm is known as Fast Iterative Shrinkage Thresholding Algorithm (FISTA) [Beck and Teboulle, 2009] and is very popular for optimizing composite functions of the form eq. (2.19). In particular, it can be proved that *FISTA* has the optimal convergence rate of $\mathcal{O}(1/k^2)$. Note that the *FISTA* algorithm is very similar to Nesterov's accelerated gradient descent for smooth functions. Basically the only difference is the application of the proximal operator in line 3.

We now mention an important point: Algorithms involving the proximal map are based on the assumption that the proximal map is "easy" to evaluate. In practice, "easy" ideally means that an explicit solution for the proximal map can be given. It turns out that this is indeed often the case for many practically relevant problems. If the proximal map cannot be solved explicitly, one has to solve a subproblem at every iteration of the algorithm. If the proximal operator is sufficiently simple, then this subproblem can be solved by a few iterations of some suitable optimization algorithm, and the overall method still converges reasonably fast.

2.1.6.3 Primal-Dual Algorithm

We now turn to completely non-smooth problems that exhibit the following structure

$$\min_{x\in\mathbb{R}^n} \{f(Kx) + g(x)\}, \tag{2.20}$$

where f, g are convex functions and K is a linear operator. This structure is very similar to the class of problems considered in the previous section, however now both functions are potentially non-differentiable.

Equation (2.20) is tackled by computing the convex conjugate of $f(Kx)$. In fact, since f is convex we have $f^{**} = f$ and eq. (2.20) can be written as the following convex-concave saddle point problem

$$\min_{x\in\mathbb{R}^n} \max_{y\in\mathbb{R}^m} \{\langle Kx, y\rangle + g(x) - f^*(y)\}, \tag{2.21}$$

where $\mathbb{R}^m$ denotes the dual space. Since the primal variable x as well as the dual variable y appear in the saddle point problem, eq. (2.21) is also called the *primal-dual formulation* of eq. (2.20). A key observation is that in the primal-dual form neither f nor g contain the linear operator K anymore. The primal-dual hybrid gradient algorithm [Chambolle and Pock, 2011] is thus obtained from the intuitive idea of taking descent steps in x and ascent steps in y. Because both f and g are non-smooth, this will be proximal gradient steps. The linear operator K can appear on either side of the scalar product through the relation $\langle Kx, y\rangle = \langle x, K^*y\rangle$, where K^* denotes the *adjoint operator*. In case we are working in a real vector space, the linear operator can be represented as a matrix with real entries and the adjoint is equivalent to the transpose of the matrix.[2] Additionally, the primal-dual algorithm uses an over-relaxation step in order to ensure convergence, see algorithm 2.5. Let $\|K\|$ denote the operator norm of the linear operator

Algorithm 2.5: Primal-dual algorithm

```
1 Initialization: Set $k = 0$, choose points $x_0, \bar{x}_0 \in \mathbb{R}^n, y_0 \in \mathbb{R}^m$, stepsizes $\tau, \sigma > 0$
2 while not converged do
3     $y_{k+1} = \text{prox}_{\sigma f^*}(y_k + \sigma K \bar{x}_k)$
4     $x_{k+1} = \text{prox}_{\tau g}(x_k - \tau K^* y_{k+1})$
5     $\bar{x}_{k+1} = 2x_{k+1} - x_k$
6     $k = k + 1$
7 end
```

K, then the algorithm converges if $\tau\sigma\|K\|^2 < 1$ with rate $\mathcal{O}(1/k)$, which is the optimal rate for non-smooth problems. As before, the basic assumption is that the proximal maps

[2]The situation is more complicated in complex vector spaces, where the adjoint is obtained from the conjugate transpose, i.e. transposing the matrix and taking the complex conjugate of the entries.

are “easy” to compute. Additionally, if g is strongly convex then the algorithm can be accelerated to rate $\mathcal{O}(1/k^2)$.

2.2 Discrete Optimization

2.2.1 Graphical Models

The basic object in discrete optimization for computer vision is the weighted (capacitated) graph. It consists of a set of nodes and a set of edges that connect the nodes. Usually the pixels of an image are the nodes, whereas the edges define the neighborhood connectivity between pixels. A probabilistic graphical model is a graph in which nodes represent random variables and edges define conditional (in)dependence assumptions. The dominant graphical models used for low-level vision tasks are undirected graphs. If additionally the conditional independence between variables is of a special form, namely such that a variable given its neighbors in the graph is conditionally independent of the rest of the graph, such graphical models are called Markov Random Fields (MRFs) [Lauritzen, 1998]. The property of local statistical independence is the *Markov property.*

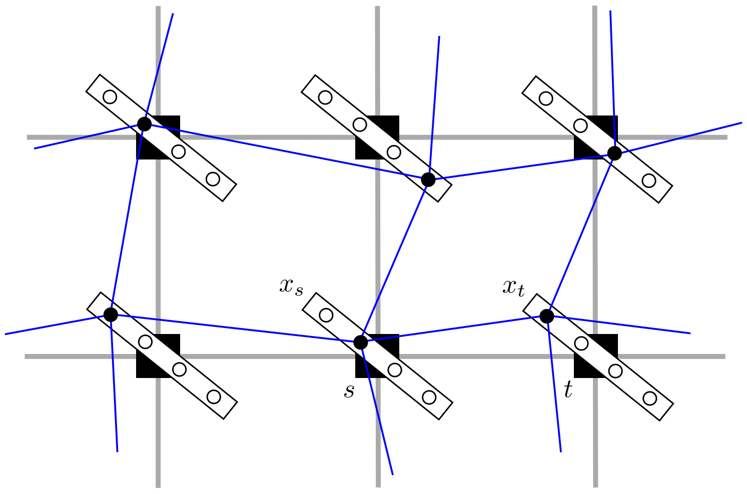


Figure 2.6: A graph on the pixelgrid. The black squares are pixels, which correspond to nodes in the graph. Each node is connected within its 4-neighborhood and carries a variable symbolized by a box with labels. The value of the label is indicated by the black circle and the labeling by blue lines.

The prototypical problem in graph-based computer vision is the labeling problem. It can be modeled by a graph in the following way, see fig. 2.6. Let $\mathcal{V}$ be a set of nodes and $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$ a set of edges, which together specify the graph $G = (\mathcal{V}, \mathcal{E})$. An edge is given as a pair $(s,t) \in \mathcal{E}$, also denoted simply st. The goal is to assign a variable x_s to each node (or pixel) $s \in \mathcal{V}$, which takes values from a discrete *labelset* $\mathcal{L} = \{l_1, \dots, l_L\}$. The value of x_s is also called the *label* at s, and in many cases the set $\mathcal{L}$ is given by the integers, i.e. $\mathcal{L} = \{0, \dots, L-1\}$. Let us define a *labeling* as the vector $x = (x_s), s \in \mathcal{V}$ of all variables.

This vector takes values in the space $\mathbf{L} = \mathcal{L}^n,\ n = |\mathcal{V}|$, the Cartesian product over all labelsets. The problem is then given as

$$\min_{x \in \mathbf{L}} E(x) = \sum_{s \in \mathcal{V}} f_s(x_s) + \sum_{st \in \mathcal{E}} f_{st}(x_s, x_t) + f_{const}. \tag{2.22}$$

The functions f are commonly referred to as *potentials*, with the first term of the energy eq. (2.22) the unary term, and the second term the binary (or pairwise) interaction potential. The last term exists for convenience to collect any constant values that might arise, e.g. from reparameterizations of the energy. The solution of eq. (2.22) corresponds naturally to the Maximum a Posteriori (MAP) configuration of a Gibbs distribution $p(x) \propto \exp(-E(x))$. Whereas higher-order interaction potentials are possible, we do not cover them here since in practice the resulting problems are much harder.

A *clique* C is a fully connected subset of nodes, i.e. $st \in \mathcal{E}$ for all $s, t \in C$. Let us also define $\mathcal{C}$, the set of all maximal cliques of the graph, i.e. cliques not properly contained in another clique. We associate to each clique a compatibility function $\varphi_c : \otimes_{s \in C} \mathcal{L}_s \to \mathbb{R}_+$, where $\otimes$ denotes the Cartesian product of the labelsets for the nodes in the clique. An important consequence of the Markov property is that the joint probability distribution $p(x_1, \ldots, x_N)$ of the graph *factorizes* as

$$p(x_1, \ldots, x_N) \propto \prod_{C \in \mathcal{C}} \varphi_C(x_C).$$

This is the basis for efficient computation of max (resp. min) marginals $p_{max}(x_s) = \max_{x', x'_s = x_s} p(x')$.

Submodularity Let x and y be two labelings, we define the operations $\wedge$ and $\vee$ as the component-wise minimum and maximum respectively

$$\begin{aligned} \forall s \in \mathcal{V} : \quad (x \wedge y)_s &= \min(x_s, y_s) \\ \forall s \in \mathcal{V} : \quad (x \vee y)_s &= \max(x_s, y_s). \end{aligned}$$

A function f is called *submodular* [Topkis, 1978, Murota, 2003], if

$$\forall x, y \in \mathbf{L} : \quad f(x \wedge y) + f(x \vee y) \leq f(x) + f(y). \tag{2.23}$$

Submodularity can be seen as the discrete equivalent to convexity for continuous functions [Lovász, 1983]. In the same way as the optimization of convex functions is relatively easy, there are powerful discrete algorithms for computing the exact minimizer of submodular functions. In particular, an energy of the form eq. (2.22) is submodular, iff all pairwise terms f_{st} are submodular functions [Boros and Hammer, 2002].

2.2.2 Exact Algorithms

The general problem eq. (2.22) is NP-hard [Boros and Hammer, 2002]. However under certain conditions, an efficient and exact solution can be computed. A standard model in computer vision is the binary image segmentation, where the labelset is given by $\mathcal{L} = \{0, 1\}$. Then the condition for submodularity eq. (2.23) takes the well-known form $f_{st}(0, 0) + f_{st}(1, 1) \leq f_{st}(0, 1) + f_{st}(1, 0)$. The binary problem can be solved by a minimum cut in polynomial time. To that end, the pairwise potentials are interpreted as edge capacities or weights. A minimum cut is a partition $\mathcal{V} = (S, T \setminus S)$ that separates the source vertex from the sink vertex and minimizes the sum of all boundary edge capacities, where a boundary edge is an edge that starts in S and ends in T. By the max-flow min-cut theorem [Elias et al., 1956, Ford and Fulkerson, 1956], the value of the minimum cut is equal to the value of the maximum flow between the source and sink vertex.

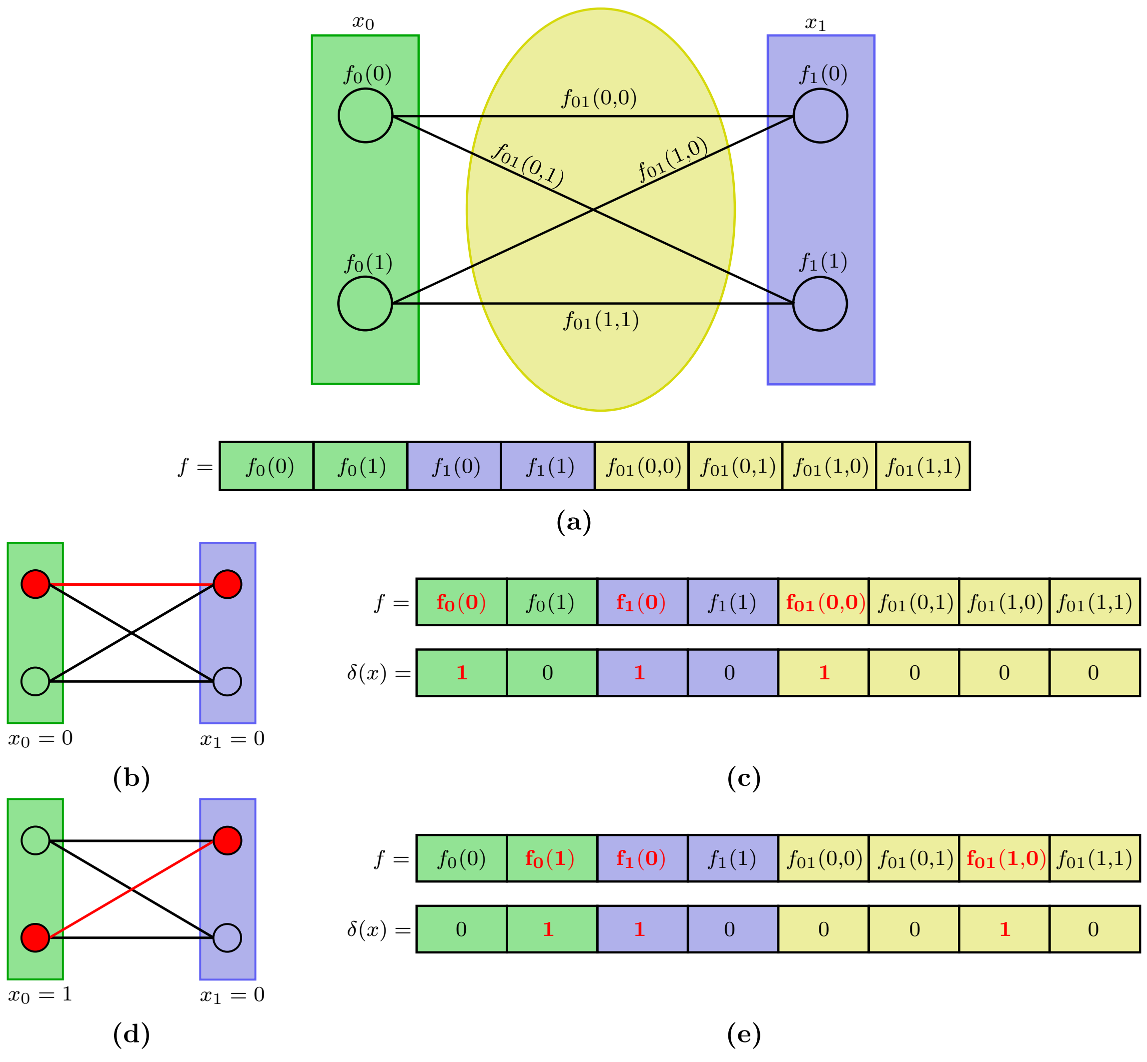
The simplest algorithm to compute the maximum flow due to [Ford and Fulkerson, 1956] works as follows. One iteratively finds paths from source to sink with non-saturated edge capacities and pushes more flow through that path until all edges of the path are saturated. If any path from source to sink passes through at least one saturated edge, the maximum flow is reached. Algorithms following this scheme are called augmenting path methods and are popular for their efficiency [Boykov and Kolmogorov, 2004]. We refer to [Cook et al., 1998] for further in-depth treatment.

Binary labeling problems, while efficiently solvable, are of limited paractical applicability. Certain multi-label problems can be reduced to a binary minimum cut with a construction proposed by [Ishikawa, 2003]. The key idea of this method is that one can use the natural ordering of the labels to construct an equivalent lifted hypergraph with binary variables. This lifted graph is then solved by a minimum cut. Such construction works in case of arbitrary unaries and submodular pairwise terms. For the special case that the unaries are convex, algorithms with better runtime bounds than Ishikawas method exist [Kolmogorov, 2005, Hochbaum, 2001].

2.2.3 Approximate Algorithms

Unfortunately many interesting energy minimization models are NP-hard. In this case methods that give approximate solutions are popular. One approach are move-making strategies [Boykov et al., 1998, Boykov et al., 2001], which try to construct an approximate global solution by a series of local binary problems. For instance, the expansion move for a label l increases the number of nodes having label l. The criterion for a local minimum is that no expansion move for any label l_i gives a labeling with lower energy. Thus in each move, the subproblem is a binary labeling problem that can be solved efficiently.

Another approach to compute approximate solutions is through the Linear Programming (LP) relaxation of the energy eq. (2.22). First, we need to introduce notation: Collect all unaries and pairwise terms in a vector $f = (f_\alpha), \alpha \in \mathcal{I}$ where the index set $\mathcal{I}$ is given as $\mathcal{I} = \{0\} \cup \{(s \in \mathcal{V}, i \in \mathcal{L})\} \cup \{(st \in \mathcal{E}, ij \in \mathcal{L} \times \mathcal{L})\}$. In this notation, $f_s(i)$ refers to the value of the unary potential at node s when x_s takes the label i. Similarly, $f_{st}(i,j)$ is the value of the pairwise term for $x_s = i, x_t = j$. We also require that $f_{st}(i,j) = f_{ts}(j,i)$, i.e. the direction does not matter.



x_0
x_1
$f_0(0)$
$f_1(0)$
$f_{01}(0,0)$
$f_{01}(0,1)$
$f_{01}(1,0)$
$f_0(1)$
$f_1(1)$
$f_{01}(1,1)$
$f =$ $f_0(0)$ $f_0(1)$ $f_1(0)$ $f_1(1)$ $f_{01}(0,0)$ $f_{01}(0,1)$ $f_{01}(1,0)$ $f_{01}(1,1)$
(a)
$x_0 = 0$
$x_1 = 0$
(b)
$f =$ $\mathbf{f_0(0)}$ $f_0(1)$ $\mathbf{f_1(0)}$ $f_1(1)$ $\mathbf{f_{01}(0,0)}$ $f_{01}(0,1)$ $f_{01}(1,0)$ $f_{01}(1,1)$
$\delta(x) =$ 1 0 1 0 1 0 0 0
(c)
$x_0 = 1$
$x_1 = 0$
(d)
$f =$ $f_0(0)$ $\mathbf{f_0(1)}$ $\mathbf{f_1(0)}$ $f_1(1)$ $f_{01}(0,0)$ $f_{01}(0,1)$ $\mathbf{f_{01}(1,0)}$ $f_{01}(1,1)$
$\delta(x) =$ 0 1 1 0 0 0 1 0
(e)

Figure 2.7: (a) A graph with two nodes, binary labels, pairwise interactions and corresponding components of the cost vector f. (b),(c) and (d),(e) show the graph and canonical overcomplete representation for the configurations $x_0 = 0, x_1 = 0$ and $x_0 = 1, x_1 = 0$ respectively.

Now we express the energy as a scalar product with the help of the mapping $\delta : \mathbf{L} \rightarrow$

$\{0,1\}^{|\mathcal{I}|}$ defined as follows

$$\begin{aligned}\delta(x)_0 &= 1\\ \delta(x)_s(i) &= [\![x_s = i]\!]\\ \delta(x)_{st}(i,j) &= [\![x_s = i, x_t = j]\!],\end{aligned}$$

where $[\![\cdot]\!]$ is the Iverson bracket, i.e. it is 1 if the argument is true and 0 otherwise. The mapping δ is called the *canonical overcomplete representation* [Wainwright and Jordan, 2008]. It is basically for each variable x_s and for each pair x_{st} an indicator of their respective label values, see fig. 2.7. The term overcomplete is justified, considering that there are lots of linear relationships between the potentials $\delta(x)$. For instance, it is easy to see that if the indicator for x_s taking the label i is true, then this induces a constraint on the pairwise potentials involving the node x_s. Namely we have that

$$\forall st \in \mathcal{E},\ \forall i \in \mathcal{L}: \quad \delta(x)_s(i) - \sum_{j' \in \mathcal{L}} \delta(x)_{st}(i,j') = 0.$$

We can now write

$$E(x) = \langle \delta(x), f \rangle = \sum_{\alpha \in \mathcal{I}} \delta(x)_\alpha f_\alpha,$$

and the energy minimization becomes

$$\min_{x \in \mathbf{L}} \langle \delta(x), f \rangle = \min_{\mu \in \mathrm{MARG}(G)} \langle \mu, f \rangle\,, \tag{2.24}$$

where MARG(G) is the so-called *marginal polytope* [Wainwright and Jordan, 2008]. It is defined as the convex hull of the vectors forming the canonical overcomplete representation, i.e. $\mathrm{MARG}(G) = \mathrm{conv}(\delta(x) \,|\, x \in \mathbf{L})$. The marginal polytope depends on the mapping δ, which in turn is defined by the structure of the graph. In general, the *LP* eq. (2.24) is not tractable, as the number of constraints in the feasible set MARG(G) grows exponentially with the problem size. Hence the *LP* is relaxed by dropping the integrality constraints of μ,[3] which is often referred to as the *standard LP relaxation* of the *MRF* energy [Živný et al., 2014].

[3]remember that the feasible set MARG(G) is constructed from the vectors δ, which are defined to be binary, i.e. $\delta(x)_s(\cdot) \in \{0,1\}\ \forall s \in \mathcal{V}$ and $\delta(x)_{st}(\cdot,\cdot) \in \{0,1\}\ \forall st \in \mathcal{E}$

Let us define the set LOCAL(G), the *local marginal polytope*, as follows

$$\mathrm{LOCAL}(G) = \left\{ \mu \in \mathbb{R}^{|\mathcal{I}|} \;\middle|\; \begin{array}{ll} \mu \geq 0 & \\ \sum_{i \in \mathcal{L}} \mu_s(i) = 1 & \forall s \in \mathcal{V} \\ \sum_{j' \in \mathcal{L}} \mu_{st}(i, j') = \mu_s(i) & \forall st \in \mathcal{E}, \forall i \in \mathcal{L} \end{array} \right\}, \tag{2.25}$$

which yields the standard *LP* relaxation of eq. (2.24)

$$\min_{\mu \in \mathrm{LOCAL}(G)} \langle \mu, f \rangle \,. \tag{2.26}$$

The first constraint in eq. (2.25) relaxes the admissable values from $\{0, 1\}$ to the positive reals. The second constraint ensures that the (fractional) label values at a node sum to 1. The third is a consistency constraint between the unary and pairwise relaxed labels. Note that LOCAL(G) is an outer bound to MARG(G), and we have

$$\min_{\mu \in \mathrm{MARG}(G)} \langle \mu, f \rangle \geq \min_{\mu \in \mathrm{LOCAL}(G)} \langle \mu, f \rangle \,,$$

whith the case of equality achieved being a tight relaxation.

The advantage of the local marginal polytope is that it contains only polynomially many constraints. However, for typical problems in computer vision this is still a large number. Standard *LP* solvers do not scale well, hence specialized algorithms have been developed.

2.2.3.1 Decomposition

The idea of decomposition methods is to split a difficult problem into smaller subproblems that are tractable, and obtain a solution by combining the individual subproblems. Consider a minimization problem of the form $\min_{x \in C} f_1(x) + f_2(x)$ where C is a convex set. Let us assume that minimizing the individual f_i is easy, but minimizing their sum is hard. For example, suppose that x is high-dimensional and decomposes as $x = (u, v, y)$, where the subvector y is low-dimensional compared to the subvectors u, v. One situation where minimizing the individual f_i is easy but minimizing the sum is hard is when the f_i almost decouple, in particular $f_1(x) = f_1(u, y)$ and $f_2(x) = f_2(v, y)$. In this situation the subvector y is called *complicating variable*, because it complicates an otherwise separable problem.

We can formulate an equivalent problem by considering local versions of the variable x and adding consistency constraints which force the local variables to be equal

$$\min_{x_1, x_2 \in C} f_1(x_1) + f_2(x_2) \quad \text{s.t.} \quad x_1 = x_2.$$

The dual decomposition is obtained by introducing Lagrange multipliers for the equality constraint, which yields the Lagrangian

$$L(x_1, x_2, \lambda) = f_1(x_1) + f_2(x_2) + \langle \lambda, x_1 - x_2 \rangle .$$

The dual subproblems are given by

$$g_1(\lambda) = \min_{x_1 \in C} f_1(x_1) + \langle \lambda, x_1 \rangle \tag{2.27}$$

$$g_2(\lambda) = \min_{x_2 \in C} f_2(x_2) - \langle \lambda, x_2 \rangle \tag{2.28}$$

and the dual problem is therefore

$$\begin{aligned} &\max_{\lambda} \{ g(\lambda) = g_1(\lambda) + g_2(\lambda) \} \qquad (2.29) \\ &= \max_{\lambda} \left\{ \min_{x_1 \in C} f_1(x_1) + \langle \lambda, x_1 \rangle + \min_{x_2 \in C} f_2(x_2) - \langle \lambda, x_2 \rangle \right\} . \end{aligned}$$

Note that the dual subproblems eqs. (2.27) and (2.28) are now the easy minimizations involving the individual f_i only, with an additional scalar offset from the Lagrange multiplier λ. Importantly, the subproblems decouple and can even be solved in parallel.

In this particular setup, eq. (2.29) is called the *master problem* and eqs. (2.27) and (2.28) are the *slave problems* [Bertsekas, 1999]. The master algorithm updates λ via the information from the solutions of the subproblems. Note that the dual problem is concave, it is composed of slave subproblems each being a minimum of a function linear in λ. Let $x_1^*(\lambda)$ be a solution for the first subproblem, i.e. $g_1(\lambda) = f_1(x_1^*(\lambda)) + \langle \lambda, x_1^*(\lambda) \rangle$. Then by concavity we have for all $\bar{\lambda}$

$$\begin{aligned} g_1(\bar{\lambda}) = \min_{x_1 \in C} f_1(x_1) + \langle \bar{\lambda}, x_1 \rangle &\leq f_1(x_1^*(\lambda)) + \langle \bar{\lambda}, x_1^*(\lambda) \rangle \\ &= f_1(x_1^*(\lambda)) + \langle \bar{\lambda}, x_1^*(\lambda) \rangle + \langle \lambda, x_1^*(\lambda) \rangle - \langle \lambda, x_1^*(\lambda) \rangle \\ &= f_1(x_1^*(\lambda)) + \langle \lambda, x_1^*(\lambda) \rangle + \langle \bar{\lambda} - \lambda, x_1^*(\lambda) \rangle \\ &= g_1(\lambda) + \langle \bar{\lambda} - \lambda, x_1^*(\lambda) \rangle , \end{aligned}$$

which shows that $g_1(\bar{\lambda}) \leq g_1(\lambda) + \langle \bar{\lambda} - \lambda, x_1^*(\lambda) \rangle$. Comparing this to the definition of the subgradient eq. (2.9), we see that the solution $x_1^*(\lambda)$ is a (sub)gradient of the dual subproblem g_1 at λ. Because one has to solve the subproblems in any case, the necessary information for updating λ in the master algorithm is obtained at essentially no additional cost through the solution of the slave problems.

Apart from the decomposition described here, the dual can be used to lower bound the original problem. Assume we have a Lagrangian $L(x_0, \lambda)$ and the problem is to compute $\min_{x_0} \max_{\lambda} L(x_0, \lambda)$. The dual problem $\max_{\lambda} \min_x L(x, \lambda)$ is a lower bound, which can be seen as follows. First we note that $L(x_0, \lambda) \geq \min_x L(x, \lambda)$, as a direct consequence of the min on the right hand side. Taking first the maximum w.r.t. λ and then the minimum w.r.t. x_0 on both sides yields

$$\begin{aligned} \max_{\lambda} L(x_0, \lambda) &\geq \max_{\lambda} \min_{x} L(x, \lambda) \\ \min_{x_0} \max_{\lambda} L(x_0, \lambda) &\geq \max_{\lambda} \min_{x} L(x, \lambda). \end{aligned}$$

Dual Decomposition Algorithms In the context of *MRF-MAP* inference, the decomposition idea is applied in the following way. We split the graph into a collection of trees T^k covering all of G, i.e. every node and every edge is part of at least one T^k. Let us also define $\mathcal{I}(T^k)$ as the set of indices corresponding to tree T^k. The trees induce a decomposition of the cost vector f into a sum $f = \sum_{k \in K} f^k$, $f^k \in \mathcal{A}^k$. Each f^k is constrained to be zero outside T^k by the constraint set $\mathcal{A}^k = \{f \in \mathbb{R}^{|\mathcal{I}|} \mid f_\alpha = 0 \ \forall \alpha \in \mathcal{I} \setminus \mathcal{I}(T^k)\}$. Then we can lower bound the *LP* eq. (2.24) as

$$\min_{x \in \mathbf{L}} \langle \delta(x), f \rangle = \min_{x \in \mathbf{L}} \left\langle \delta(x), \sum_k f^k \right\rangle \geq \sum_k \min_{x \in \mathbf{L}} \left\langle \delta(x), f^k \right\rangle. \tag{2.30}$$

However, the decomposition $f = \sum_k f^k$ is not unique, and it is natural to consider the question which decomposition gives the tightest bound. This amounts to maximizing the lower bound eq. (2.30) over all possible decompositions

$$\max_{f^k \in \mathcal{A}^k, \sum_k f^k = f} \sum_k \min_{x \in \mathbf{L}} \left\langle \delta(x), f^k \right\rangle. \tag{2.31}$$

A fundamental theorem ([Wainwright et al., 2005], Theorem 1) links this maximization, which is a *LP*, to the standard *LP* relaxation of the *MRF* energy. In particular, it turns out that the *LP* relaxation eq. (2.26) is the Lagrangian dual to eq. (2.31). An important consequence of this theorem is that the optimal value of eq. (2.31) does not depend on the choice of decomposition, as long as the trees cover the whole graph.

For images, where the graph is given by the pixelgrid, a popular choice is a decomposition into horizontal and vertical chains. Such chains are can be solved efficiently by e.g. dynamic programming. In view of the decomposition approach, the chains correspond to the slave problems eqs. (2.27) and (2.28) and the outer maximization in eq. (2.31) is the master problem. Compared to the equivalent *LP* relaxation eq. (2.26), the problem eq. (2.31) has orders of magnitude less constraints thanks to the decomposition. However, it is still a large-scale optimization problem.

There are different approaches for optimizing the master problem. A straightforward method is the subgradient algorithm [Komodakis et al., 2011, Kappes et al., 2012], which guarantees convergence to the optimal value of the relaxed dual. Sequential Tree-Reweighted Message Passing (TRW-S) [Kolmogorov, 2006] is an efficient block-coordinate ascent algorithm based on dynamic programming, however this method might get stuck in suboptimal fixed points. A massively parallel algorithm is Dual Minorize-Maximize (DMM) [Shekhovtsov et al., 2016], which aims specifically at exploiting the speedup enabled by parallel capabilities of programmable Graphics Processing Units (GPUs). The idea of this method is to alternate between computing minorants of the slave problems and using these minorants to update (maximize) the master problem – hence the name *DMM*. The key property is that computing the minorants can be implemented very efficiently in a parallel manner. *DMM* has the same fixed points as *TRW-S*, but is much faster in practice.

2.3 Differential Geometry

The material in this section is based on the two classic textbooks [Spivak, 1999, do Carmo, 1992], as well as on the extensive book by [Lee, 2012]. The geometric approach to tensors is inspired by the enlightening script [Grinfeld, 2013].

The topic of differential geometry is the study of geometric structures by means of calculus. For example, given a curve in the two-dimensional Euclidean plane, one could ask about the *tangent* to the curve. The tangent at a fixed point P on the curve can be found as follows:

> Choose a point $\hat{P}$ near P on the curve and construct the unique straight line passing through the two points. Let $\hat{P}$ come closer to P, the tangent is the line obtained from the limit of this procedure.

Whereas in modern terminology the tangent is defined by the *gradient* of the curve at P, we point out that this recipe does not involve coordinates or functions. It uses only the geometric concepts of distance, points and lines in the plane.

Geometry is one of the oldest branches of mathematics with a history of more than two millenia, rooted in practical problems of construction, measurements of land and the motion of stars. These tasks were solved by classical compass-and-straightedge constructions, in which the main elements are the notion of points, lines, circles, length and angle. Many of the mathematical methods can be traced back to the cultures of Babylonia and Egypt, which were then refined further by the ancient Greeks. This development eventually culminated in the great work of Euclid of Alexandria, *The Elements* (ca. 300 B.C.). In this script, which is regarded as one of the most important works in the history of mathematics, Euclid introduced what is today known as the *axiomatic method*: Starting from a small set of axioms, propositions are developed by careful logical reasoning, eventually giving a complete and consistent picture of the mathematics known at the time. Up to this day, geometry obeying the axioms laid down by Euclid is known as *Euclidean geometry*.

We emphasize the absence of coordinates, vectors and algebraic operations in the classical geometry described in *The Elements*. All propositions and proofs are given in terms of intuitive geometric concepts like lines, length and angles. Indeed, from a historic perspective the invention of coordinate systems is quite new. Attributed to René Descartes in the seventeenth century, a coordinate system systematically assigns a set of numbers to points in space. This enabled the use of algebraic methods for solving geometric problems, an event that marked a turning point in the history of science. A famous quote by Joseph Lagrange described it as follows:

> As long as algebra and geometry have been separated, their progress has been slow and their uses limited, but when these two sciences have been united, they have lent each mutual forces, and have marched together towards perfection.

The "mutual forces" of geometry and algebra ultimately led to the development of calculus, and around the year 1700 scientific advancement happened at an unprecedented rate of progress. Newton and Leibniz developed differential calculus, Kepler, Descartes, Fermat and others found new algebraic space curves, members of the Bernoulli family are linked with the solution of many ordinary differential equations, and Euler and Lagrange set up the foundations of the calculus of variations.

With differential calculus readily available, great success has been made in adapting geometric concepts such as distance between points, area and angles to arbitrary curves and surfaces. A prime example of a geometric problem that was solved by calculus is the task of finding a surface of minimum area with given boundary, i.e. minimizing the surface area

$$\min_{\mathcal{S}} A(\mathcal{S}) = \int_{\mathcal{S}} \mathrm{d}s \tag{2.32}$$

over all possible surfaces $\mathcal{S}$ with fixed boundary. This problem has various connections to real phenomena, for example the surface spanned by a soap film across a fixed boundary takes the form of a minimal surface, see fig. 2.8. Euler and Lagrange studied such problems in the context of the calculus of variations, and Euler is credited with discovering a formal description of a non-trivial example of a minimal surface: The catenoid. Lagrange found the differential equation of the catenoid for a surface of the form $z = f(x, y)$, with area element $\mathrm{d}s = \sqrt{1 + |\nabla f|^2} = \sqrt{1 + f_x^2 + f_y^2}$. He considered critical points of the variational problem eq. (2.32) by computing the associated Euler-Lagrange equation. With the Lagrangian $L(x, y, f, \nabla f) = \sqrt{1 + |\nabla f|^2}$, the Euler-Lagrange equation is given by

$$\frac{\partial L}{\partial f} - \operatorname{div}\left(\frac{\partial L}{\partial \nabla f}\right) = 0. \tag{2.33}$$

Inserting $\frac{\partial L}{\partial f} = 0$ and

$$\frac{\partial L}{\partial \nabla f} = \left[\tfrac{f_x}{\sqrt{1+|\nabla f|^2}}, \tfrac{f_y}{\sqrt{1+|\nabla f|^2}} \right]^T$$
$$\operatorname{div}\left(\frac{\partial L}{\partial \nabla f}\right) = \frac{f_{xx}(1+|\nabla f|^2) - f_x(f_{xx}f_x + f_{xy}f_y) + f_{yy}(1+|\nabla f|^2) - f_y(f_{yy}f_y + f_{xy}f_x)}{(1+|\nabla f|^2)^{3/2}}$$

into eq. (2.33) yields the following differential equation (due to Lagrange) for the catenoid:

$$f_{xx}(1+f_y^2) + f_{yy}(1+f_x^2) - 2f_{xy}f_xf_y = 0. \tag{2.34}$$

It is interesting that neither Euler nor Lagrange realized the geometric meaning of their equations. Later it was proved by Meusnier that eq. (2.34) is equivalent to the vanishing of the mean curvature on the surface. Meusniers work is the basis for the modern definition of a minimal surface: *A surface $\mathcal{S}$ is minimal if its mean curvature H is zero.* In the special case that the surface is given as $z = f(x, y)$, the mean curvature reads

$$H = \frac{1}{2}\frac{f_{xx}(1+f_y^2) + f_{yy}(1+f_x^2) - 2f_{xy}f_xf_y}{(1+|\nabla f|^2)^{3/2}}.$$

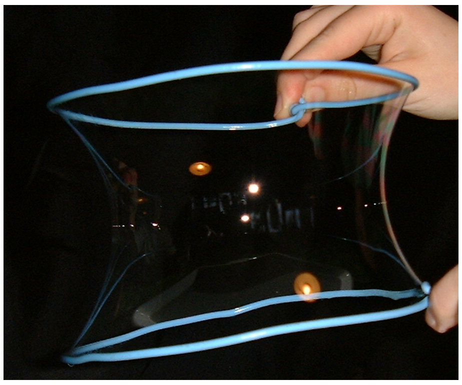


Figure 2.8: A soap film forms a catenoid (Image source: wikipedia, CC0 public domain)

2.3.1 Coordinate Systems

One of the reasons Lagranges differential equation for the catenoid is hard to interpret geometrically is that we silently introduced a coordinate system by specifying the surface as the graph of a function $z = f(x, y)$. Therefore eq. (2.34) not only carries information about the algebraic solution of the variational minimal surface problem, it also inherently carries information about the coordinate system being used. In the example above, we employed a *Cartesian* coordinate system. A Cartesian coordinate system is characterized by the fact that the basis vectors are orthonormal.

Although Cartesian coordinates are arguably the most natural and most widely used, it is important to realize that the choice of coordinate system in general is not fixed. A given problem may be easier to analyze in one coordinate system than in another. Finding the differential equation for the catenoid is best done using Cartesian coordinates, it would be much harder were the problem formulated e.g. in spherical coordinates. On the other hand, consider the curve depicted in fig. 2.9. It would be a daring task to give the curve equation in Cartesian coordinates, but in polar coordinates it is simple: The curve equation reads $r(\varphi) = \sin(2^{\varphi}) - 2.2$.

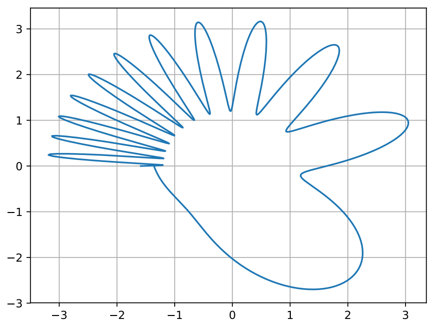


Figure 2.9: The curve with equation $r(\varphi) = \sin(2^{\varphi}) - 2.2$ in polar coordinates would be very complicated to express in Cartesian coordinates.

We see that in order to investigate the geometry of curves and surfaces by means of calculus, we have to introduce a coordinate system. But it should not stand in the way of the perception of geometric properties. A coordinate system is merely a useful tool, nothing more. In particular, we wish that geometric properties of a curve or surface evaluate the same in *any* coordinate system. In fact, by definition a property is *geometric*, if it is independent from the coordinate system.

We thus begin with the following very simple definition of a vector:

Definition 2.10. A vector $\mathbf{V}$ is a finite, directed line segment.

Vectors live in a Euclidean space, since only the Euclidean space can accommodate straight lines. Note that this definition is geometric, it does not involve coordinates. In fact, it is counterproductive to think of $\mathbf{V}$ as a n-tuple of components – we haven't introduced a coordinate system with respect to which the components could be given. We will use boldface notation whenever we refer to vectors as geometric objects according to definition 2.10. Vectors can be naturally added together (for example by the tip-to-tail rule) and scaled by a scalar, which makes Euclidean space into a vector space, usually denoted by $\mathbb{E}^n$. In principle all of the following results hold in any dimension n, but since spaces with $n > 3$ are hard to visualize we will usually give examples in a two- or three-dimensional space.

What can we do with vectors? Since a vector is a *finite* line segment, its length, denoted $|\mathbf{V}|$, can be taken. We emphasize that in this context length is a primary concept that is not defined in terms of anything else.

A very important and useful operation is the dot product, which is defined as follows:

Definition 2.11. The dot product of vectors $\mathbf{U}$ and $\mathbf{V}$ is given by

$$\mathbf{U} \cdot \mathbf{V} = |\mathbf{U}||\mathbf{V}| \cos \alpha,$$

where α is the angle between $\mathbf{U}$ and $\mathbf{V}$. The dot product is bilinear and symmetric.

While this definition might seem quite arbitrary from an algebraic point of view, it turns out that the dot product has a lot of nice geometric properties. For example, it follows directly from definition 2.11 that

- $\mathbf{U} \cdot \mathbf{U} = |\mathbf{U}|^2$, the dot product of a vector with itself is its length squared.
- $\mathbf{U} \cdot \mathbf{V} = 0 \Leftrightarrow \mathbf{U} \perp \mathbf{V}$, if the dot product of two (nonzero) vectors is zero, the vectors are orthogonal.

We can find the *orthogonal projection* of $\mathbf{V}$ onto $\mathbf{U}$, denoted by $\mathrm{proj}_{\mathbf{U}}(\mathbf{V})$, with the help of the dot product as follows (see fig. 2.10(b))

$$\mathrm{proj}_{\mathbf{U}}(\mathbf{V}) = \frac{\mathbf{U} \cdot \mathbf{V}}{\mathbf{U} \cdot \mathbf{U}} \mathbf{U}.$$

We emphasize again that the definition of the dot product is purely geometric. This is possible, because the length of a vector is a primary concept. The angle α that appears in definition 2.11 is used only for convenience: Alternatively, the dot product can be defined solely in terms of length, without introducing angles

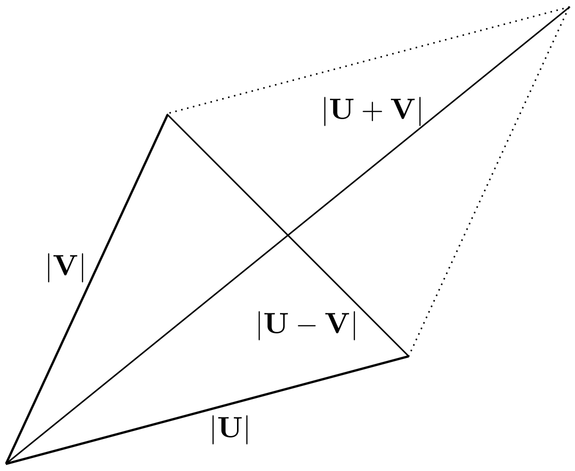
$$\mathbf{U} \cdot \mathbf{V} = \frac{|\mathbf{U} + \mathbf{V}|^2 - |\mathbf{U} - \mathbf{V}|^2}{4}. \tag{2.35}$$

This can be derived by considering the squared length of the sum of the two vectors

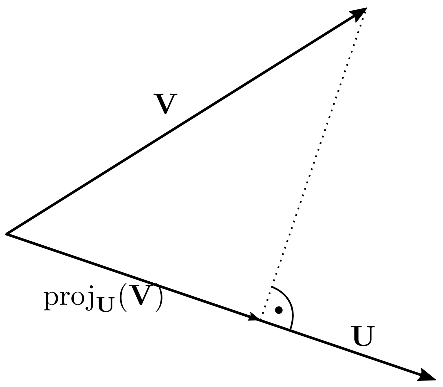
$$\begin{aligned} |\mathbf{U} + \mathbf{V}|^2 &= (\mathbf{U} + \mathbf{V}) \cdot (\mathbf{U} + \mathbf{V}) \\ &= \mathbf{U} \cdot \mathbf{U} + 2\mathbf{U} \cdot \mathbf{V} + \mathbf{V} \cdot \mathbf{V}, \end{aligned}$$

where we used linearity of the dot product to expand the expression. From this, we get the identity $2\mathbf{U} \cdot \mathbf{V} = |\mathbf{U} + \mathbf{V}|^2 - \mathbf{U} \cdot \mathbf{U} - \mathbf{V} \cdot \mathbf{V}$. If we do the same for the difference between the vectors, we get $2\mathbf{U} \cdot \mathbf{V} = \mathbf{U} \cdot \mathbf{U} + \mathbf{V} \cdot \mathbf{V} - |\mathbf{U} - \mathbf{V}|^2$. Summing the two identities yields eq. (2.35). In geometry, the relation we have used is known as the parallelogram law, see fig. 2.10(a). It states that the sum of the squared lengths of the sides of a parallelogram is equal to the sum of the squared lengths of the diagonals. Having a definition of the dot product in terms of length only, we point out another geometric property of the dot product

- $\cos\alpha = \frac{\mathbf{U}\cdot\mathbf{V}}{\sqrt{\mathbf{U}\cdot\mathbf{U}}\sqrt{\mathbf{V}\cdot\mathbf{V}}}$, the dot product can be used to define the concept of angle between vectors.



(a) The dot product between **U** and **V** can be computed geometrically via the parallelogram law $2|\mathbf{U}|^2 + 2|\mathbf{V}|^2 = |\mathbf{U}+\mathbf{V}|^2 + |\mathbf{U}-\mathbf{V}|^2$.



(b) Orthogonal projection of **V** onto **U**.

Figure 2.10: Parallelogram law and orthogonal projection.

2.3.1.1 $\mathbb{R}^n$ and Euclidean Space

In order to formally establish the notion of a coordinate system, we first need to introduce the space $\mathbb{R}^n$. Again, we will mostly use $n = 2, 3$ for examples, but the theory carries in an obvious way to arbitrary dimensions.

Definition 2.12. The space $\mathbb{R}^n$ is the set of all ordered n-tuples of real numbers. A n-tuple $x = (x^1, \ldots, x^n)$ is called a *point* of $\mathbb{R}^n$, and the numbers $x^1, \ldots, x^n \in \mathbb{R}$ are called the *components* of x.

Following the usual conventions of differential geometry, we use superscripts to denote the components of points. Note that there co-exist different possibilities how to interpret the space $\mathbb{R}^n$. The first interpretation is merely as a topological space according to definition 2.12. At the same time, $\mathbb{R}^n$ can be considered as a vector space, with component-wise addition and scalar multiplication of points. This is arguably the most common use of $\mathbb{R}^n$.

While this might seem trivial, it turns out that precise treatment of these matters is crucial for the modern formulation of calculus on manifolds and Riemannian geometry in general. In fact, by a basic theorem of linear algebra any two vector spaces of the same dimension are isomorphic, i.e. there exists a bijective, invertible mapping between them. However, in general they are not *canonically isomorphic*, meaning the isomorphism depends on the choice of basis in the two vector spaces. For example, the plane P of all points orthogonal to $x = (1,1,1)$ is clearly a two-dimensional subspace of $\mathbb{R}^3$. Hence P is isomorphic to $\mathbb{R}^2$, but it is not canonically isomorphic. The isomorphism $(x^1, x^2, 0) \mapsto (x^1, x^2)$ is one obvious example among many, and it depends on the way in which we represent elements of P by means of a basis.

In general, one can always represent elements of a n-dimensional vector space by a n-tuple of numbers, i.e. as points of $\mathbb{R}^n$. These numbers are the coefficients of a linear combination of basis vectors, and the resulting n-tuple is called the coordinate or component representation of the vector. Critically, the vector itself and its component representation are different things. *Any* set of elements that can be added together and multiplied by a scalar forms a vector space. Take for example the vector space $\mathbb{P}_3$ of polynomials of degree ≤ 2. Clearly $p = 3x^2 - 2$ is a vector in this space. Using the basis $\{x^2, x, 1\}$, the coordinate representation of p is the triplet $(3, 0, -2)$. We see that the vector and its component representation are fundamentally different, for instance a triplet of numbers cannot be integrated or differentiated, whereas a polynomial can. While it certainly makes sense to call elements of a vector space "vectors", this terminology is a bit unfortunate. First, it collides with the concept of a geometric vector from the previous section, definition 2.10. Second, the most simple case of the vector space $\mathbb{R}^n$ is also the most confusing. Both, vectors in $\mathbb{R}^n$ and their coordinate representation, are tuples of numbers. This makes it easy to conflate the two concepts. For this reason, we started our presentation with geometric vectors and made a point not to think about them as tuples of numbers.

$\mathbb{R}^n$ as a vector space with component-wise addition and scalar multiplication of points is quite special: it is canonically isomorphic to $\mathbb{R}^n$ the topological space, i.e. it exhibits a *canonical* (or *natural, standard*) basis. This means that the basis in a sense is given by the nature of the space itself. For instance, in $\mathbb{R}^3$ with component-wise addition and scalar multiplication of points, we have the situation that the elements of the space (the vectors) are themselves triplets of numbers. There is a natural basis, characterized by the fact that any vector $x \in \mathbb{R}^3$ is equal to its triplet of coordinates with respect to the natural basis. That basis is of course the set $\{e_1 = (1,0,0), e_2 = (0,1,0), e_3 = (0,0,1)\}$, and the representation of a vector is obtained via $x = \sum_{i=1}^{3} x^i e_i$.

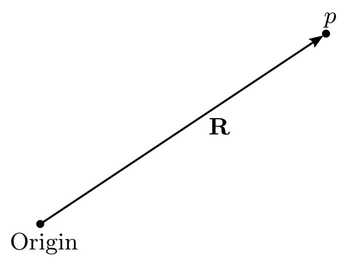


Figure 2.11: The position vector $\mathbf{R}$ is the directed line segment from the origin to a point in Euclidean space.

This leads to yet another interpretation of $\mathbb{R}^n$, namely as a model for the Euclidean space $\mathbb{E}^n$. In fact, this is the interpretation most closely related to the notion of geometric vectors that we have encountered in section 2.3.1. Consider a two-dimensional plane, we introduce the position vector $\mathbf{R} \in \mathbb{E}^2$ which represents points in the Euclidean space with respect to some arbitrary origin, see fig. 2.11. We then *identify* the position vector $\mathbf{R}$ with a vector[4] $r \in \mathbb{R}^2$, where r is given by a pair of *coordinates*, $r = (r^1, r^2)$. Renaming the coordinates r^1, r^2 to x, y, one obtains the well-known presentation of elementary geometry often taught in high school. It is very intuitive and appealing, hence it is no surprise that this identification propelled the "mutual forces of geometry and algebra". In fact it is so powerful that oftentimes the identification is assumed silently, and $\mathbb{R}^n$ is simply equated with $\mathbb{E}^n$. This approach however can obstruct understanding of the modern notion of manifolds and the role of coordinates.

Let us therefore mention explicitly the concepts involved in identifying $\mathbb{E}^n$ with $\mathbb{R}^n$. First, we are using $\mathbb{R}^n$ with its vector space structure. Geometry starts with the measurement of length, therefore we need to carry the notion of length over to $\mathbb{R}^n$. A standard way to achieve this is to endow $\mathbb{R}^n$ with the structure of an inner product. We point out that whereas a norm on $\mathbb{R}^n$ would also provide a notion of length, an inner product provides more structure than a norm. In particular, a norm is not enough to do geometry, we need to have the structure of an inner product.

Definition 2.13. An inner product on a real vector space V is a mapping $\langle \cdot, \cdot \rangle : V \times V \to \mathbb{R}$ which satisfies for $v, w, q \in V$ and $c \in \mathbb{R}$

1. Symmetry
$$\langle v, w \rangle = \langle w, v \rangle \,;$$

2. Bilinearity
$$\begin{aligned} \langle v + q, w \rangle &= \langle v, w \rangle + \langle q, w \rangle \,, \\ \langle v, w + q \rangle &= \langle v, w \rangle + \langle v, q \rangle \,, \\ \langle cv, w \rangle &= c \langle v, w \rangle = \langle v, cw \rangle \,; \end{aligned}$$

3. Positive Definiteness
$$\langle v, v \rangle \geq 0,$$
with equality iff $v = 0$.

An inner product induces a *norm* or *length* of a vector, denoted $\|v\|$, via $\|v\| = \sqrt{\langle v, v \rangle}$. It is easy to see that this expression fulfills the requirements of a norm. A vector space with an inner product is called inner product space. Note that whereas every inner product

[4] here we mean "vector" as an element of the vector space, not a geometric vector.

space is also a normed space, the reverse is not true. In general there is no canonical way how to choose an inner product for a vector space. For example, it is not hard to see that the mapping $\langle v, w\rangle = 3v^1w^1 + 2v^2w^2$ fulfills the requirements of definition 2.13 and is therefore a valid inner product on $\mathbb{R}^2$.

As a short detour, we now show that one of the most famous geometric theorems, the Pythagorean theorem, follows directly from the structure of an inner product. In an inner product space we say that vectors u, v are orthogonal if their inner product is zero: $\langle u, v\rangle = 0 \Leftrightarrow u \perp v$.

Lemma 2.4 (Pythagorean theorem)**.** *Let $u, v \in \mathbb{R}^n$ be orthogonal vectors. Then* $\|u+v\|^2 = \|u\|^2 + \|v\|^2$.

Proof. Expressing the squared norm as an inner product and using linearity of the inner product and the fact that $\langle u, v\rangle = 0$, we find

$$\|u+v\|^2 = \langle u+v, u+v\rangle = \langle u, u\rangle + 2\langle u, v\rangle + \langle v, v\rangle = \|u\|^2 + \|v\|^2.$$

$\square$

In the same way that the vector space $\mathbb{R}^n$ is special in that it has a natural basis, it also has a natural inner product, given by $\langle v, w\rangle = \sum_i v^i w^i$. We can characterize this inner product by the fact that the natural basis with respect to the natural inner product is orthonormal

$$\langle e_i, e_j\rangle = \begin{cases} 1 \text{ if } i = j \\ 0 \text{ else} \end{cases}.$$

Moreover, it turns out that the natural inner product behaves geometrically in the same way as the dot product in Euclidean spaces that we have given in definition 2.11! This is a surprising fact, since the dot product for geometric vectors is a quite arbitrary algebraic expression that just happens to yield useful geometric relations. Due to this connection to the dot product, the natural inner product is also called Euclidean inner product or standard inner product. For the purpose of identifying $\mathbb{E}^n$ with $\mathbb{R}^n$ this is useful, since it allows to carry over all the geometric concepts related to the dot product to the vector space $\mathbb{R}^n$.

We point out the following observation: Even though the dot product on $\mathbb{E}^n$ and the standard inner product on $\mathbb{R}^n$ are equivalent in their behavior, the reasoning behind the two approaches is in a sense opposite. In the geometric space $\mathbb{E}^n$, the length of a vector is a given concept that exists a priori. The dot product, and consequently other geometric properties like orthogonality, are derived from the concept of length. In the vector space $\mathbb{R}^n$, it is the other way around: The structure of an inner product is defined axiomatically, and length and other relations are derived from an (any!) inner product.

The last piece missing for the identification are *coordinates.* In elementary analytic geometry, one usually starts by investigating objects in the two-dimensional Euclidean plane. This is done by means of congruency of triangles, length of line elements, angles between lines etc. Such approach is in the spirit of Euclid and the geometers before the 18^{th} century, who certainly did not think of geometric vectors as tuples of numbers. Later, coordinates are introduced through the following device: Choose mutually orthogonal "number axes" and define a "coordinate mapping" between $\mathbb{E}^2$ and $\mathbb{R}^2$ by

$$\begin{aligned} \varphi \ : \ & \mathbb{E}^2 \to \mathbb{R}^2 \\ & \mathbf{R} \mapsto (x^1(\mathbf{R}), x^2(\mathbf{R})), \end{aligned} \tag{2.36}$$

with (x^1, x^2) the coordinates of the point represented by the position vector $\mathbf{R}$. Note that we think of coordinates as *functions.* We require that the mapping φ be a homeomorphism in order to avoid degenerate configurations. A homeomorphism is a bijective, continuous mapping that admits a continuous inverse. This requirement makes immediate sense, since we wish for the coordinate mapping to map all points in a unique way, i.e. surjective and injective, thus bijective.

With such a mapping, we can establish further correspondence of elementary geometric objects. For example, lines in $\mathbb{E}^2$ correspond to subsets of $\mathbb{R}^2$ consisting of the solutions of linear equations. While the mapping φ makes the identification possible, it is crucial to note that φ *is chosen arbitrarily.* There is no natural, geometric way to identify the two spaces. We have seen at the beginning of this chapter that the choice of coordinates is rather dictated by the problem at hand. We conclude that $\mathbb{E}^n$ may be identified with an inner product space $\mathbb{R}^n$ plus a coordinate system. Clearly this is very practical, since coordinates can serve as a powerful computational tool. Therefore we usually want to make the identification, however we should always keep in mind that an arbitrary choice of coordinate system is involved.

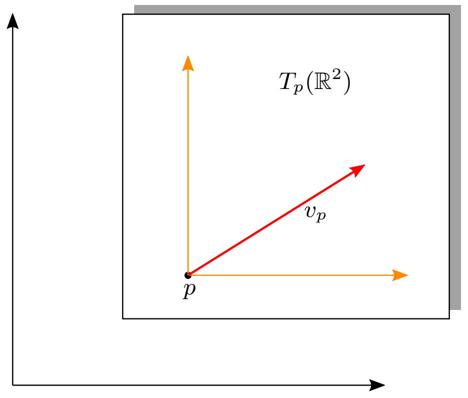
2.3.2 Tangent Vectors

In the previous chapter we have already hinted that there is an overloading of the meaning of "vector", which we would like to clarify now. On the one hand we think of vectors simply as elements of a vector space. In the case of $\mathbb{R}^n$ these are points in space, described by their coordinates $(x^1, \ldots, x^n)$. On the other hand, we also think about vectors as directed line segments in the sense of definition 2.10, i.e. as geometric vectors. Whereas a point in space is completely determined by its position, a geometric vector consists of two pieces of information: The start point p and the "vector part" v. We visualize the geometric vector as the arrow from p to $p + v$. When we talk about vectors as arrows, we often focus exclusively on the vector part v, disregarding the start point p. As we will see, this is only possible in the special case of $\mathbb{R}^n$ and a Cartesian coordinate system. We emphasize that in general vectors with the same vector part v but different start points

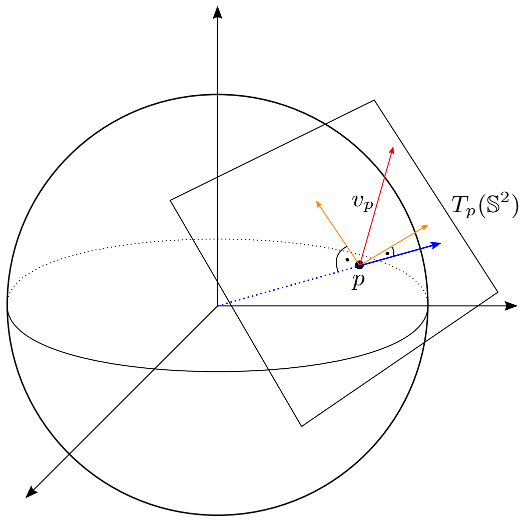
p are *different objects*. For example, in general is not possible to add two vectors unless their start point is the same: Consider classical mechanics, where a force vector acts on some object. Clearly it makes a difference *where* the force is applied to the object. Hence we would like to formalize the notion of “arrows attached to a point”.

On a different note, we recall that the central idea of calculus is linear approximation. As stated at the beginning of this chapter, differential geometry is the study of geometric structures by means of calculus. In order to generalize the ideas of calculus to arbitrary manifolds, we need some sort of “linear model” of the manifold at a point.

We thus introduce the concept of a *tangent space* to $\mathbb{R}^n$ at a point p, denoted by $T_p(\mathbb{R}^n)$, as the set of all arrows emanating from p. For shorter notation, we may omit the parentheses and write $T_p\mathbb{R}^n$ for $T_p(\mathbb{R}^n)$. An element of the tangent space is called tangent vector (or simply vector) of $\mathbb{R}^n$ at p and denoted by v_p. Intuitively, just as the tangent to a curve is the line that “just touches” the curve at a point p, the tangent space “touches” $\mathbb{R}^n$ at p. The tangent space $T_p(\mathbb{R}^n)$ is of course again just $\mathbb{R}^n$ with the origin shifted to the point p, see fig. 2.12(a). If we consider a more interesting example, say the sphere $\mathbb{S}^2$, we could think of its tangent space at a point p intuitively as the plane that “touches” the sphere at p as depicted in fig. 2.12(b). We could define this formally as the subspace of $T_p(\mathbb{R}^3)$ of vectors orthogonal[5] to the radial vector through p. This approach however presupposes that the sphere is embedded in 3-dimensional Euclidean space and it does not work for an arbitrary manifold where there might not be an ambient space.



(a) The tangent space of $\mathbb{R}^2$.



(b) The tangent space of the sphere $\mathbb{S}^2$, with the radial vector in blue.

Figure 2.12: The tangent space is a “linear model” of the space at a point p. Basis vectors of the tangent space are depicted in orange, the tangent vector in red. Figure adapted from [Lee, 2012].

[5]orthogonality uses the inner product that $T_p\mathbb{R}^3$ inherits from $\mathbb{R}^3$ through the natural isomorphism $\mathbb{R}^3 \cong T_p\mathbb{R}^3$.

Therefore another characterization of tangent vectors is needed, one that does not depend on the ambient space at all. The standard approach in modern differential geometry is through the concept of *derivations* of smooth functions.

We denote by $C^\infty(\mathbb{R}^n)$ the set of all real-valued, smooth functions on $\mathbb{R}^n$, i.e. functions $f : \mathbb{R}^n \to \mathbb{R}$ with continuous partial derivatives of all orders. A tangent vector v_p can be used to take the directional derivative of a function $f \in C^\infty(\mathbb{R}^n)$ at a point $p \in \mathbb{R}^n$

$$D_v f|_p = D_v f(p) = \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} f(p + tv), \tag{2.37}$$

which is just the usual rate of change of f in direction v at the point p. If we write the tangent vector v_p in terms of the standard basis as $v_p = \sum_i v^i e_i$, by the chain rule the directional derivative becomes

$$D_v f|_p = \sum_i \left.\frac{\mathrm{d}}{\mathrm{d}t}\right|_{t=0} (p^i + tv^i) \frac{\partial f}{\partial x^i}(p) = \sum_i v^i \frac{\partial f}{\partial x^i}(p). \tag{2.38}$$

To avoid clutter, we may omit the point of application and write $D_v f$ if the meaning is clear from the context.

Let w denote a map from the space of smooth function to the reals, i.e. $w : C^\infty(\mathbb{R}^n) \to \mathbb{R}$. The map w is called *derivation at the point* p if it is linear and satisfies the product rule

$$w(fg) = g(p)wf + f(p)wg \tag{2.39}$$

for $f, g \in C^\infty(\mathbb{R}^n)$ and $p \in \mathbb{R}^n$. Interpreting the directional derivative as an operator on smooth functions, any tangent vector v_p gives rise to a map $D_v|_p : C^\infty(\mathbb{R}^n) \to \mathbb{R}$ by means of eq. (2.38). It is easy to see that $D_v|_p$ is a derivation, since the partial derivatives $\frac{\partial f}{\partial x^i}(p)$ possess the required properties, i.e. the differentiation operator ∂ is linear and fulfills eq. (2.39) [Werner, 2007]. Hence there is a map from the tangent space to the space of derivations $\mathfrak{D}_p(\mathbb{R}^n)$ at p

$$\begin{aligned} \phi : T_p(\mathbb{R}^n) &\to \mathfrak{D}_p(\mathbb{R}^n) \\ v_p &\mapsto D_v|_p = \sum_i v^i \frac{\partial f}{\partial x^i}(p). \end{aligned} \tag{2.40}$$

We now consider in more detail the space of derivations $\mathfrak{D}_p(\mathbb{R}^n)$. This is a vector space on its own, with the operations

$$(w_1 + w_2)f = w_1 f + w_2 f, \quad (cw)f = c(wf),$$

for $w, w_1, w_2 \in \mathfrak{D}_p(\mathbb{R}^n)$, $f \in C^\infty(\mathbb{R}^n)$ and $c \in \mathbb{R}^n$.

One of the fundamental observations in differential geometry is that there is a one-to-one correspondence between tangent vectors and derivations at p. In fact, we are going to prove the (somewhat unexpected) fact that the vector space $\mathfrak{D}_p(\mathbb{R}^n)$ is finite-dimensional and naturally isomorphic to the tangent space $T_p(\mathbb{R}^n)$. To that end, we will need two additional lemmas.

Lemma 2.5. *Let $p \in \mathbb{R}^n$, $w \in \mathfrak{D}_p(\mathbb{R}^n)$ and $f \in C^\infty(\mathbb{R}^n)$. If f is a constant function, then $wf = 0$.*

Proof. Since we do not know if every derivation is also a directional derivative, we can use only the defining properties of a derivation. By linearity of derivations, $w(cf) = cwf$, with $c \in \mathbb{R}$. Therefore it suffices to prove the claim for the constant function $f_1 \equiv 1$, since wf for any $f \equiv c$ can then be written as $c(wf_1)$. By the product rule eq. (2.39), we have for $f_1 \equiv 1$

$$w(1) = w(1 \cdot 1) = 1 \cdot w(1) + w(1) \cdot 1 = 2w(1).$$

Subtracting $w(1)$ from both sides gives the desired result $w(1) = 0$. □

This lemma simply asserts the well-known fact that the derivative (derivation) of a constant function is zero.

Lemma 2.6 (Taylor's theorem)**.** *Let $U \subset \mathbb{R}^n$ be a convex subset of $\mathbb{R}^n$, $f \in C^\infty(U)$ and fix a point $p \in U$. Then there are functions $g_1(x), \dots, g_n(x) \in C^\infty(U)$ such that for any $x \in U$*

$$f(x) = f(p) + \sum_i (x^i - p^i) g_i(x), \quad g_i(p) = \frac{\partial f}{\partial x^i}(p).$$

Proof. With U being convex, the line segment $p + t(x-p)$, $t \in [0,1]$ lies in U for all $x \in U$. Hence, $f(p + t(x-p))$ is defined for $0 \le t \le 1$. By the chain rule, we have that $\frac{\mathrm{d}}{\mathrm{d}t} f(p + t(x-p)) = \sum_i (x^i - p^i) \frac{\partial f}{\partial x^i}(p + t(x-p))$. Integrating both sides w.r.t. t from 0 to 1, we obtain

$$f(p + t(x-p))\Big|_0^1 = \sum_i (x^i - p^i) \int_0^1 \frac{\partial f}{\partial x^i}(p + t(x-p)) \mathrm{d}t. \tag{2.41}$$

Now we let $g_i(x) = \int_0^1 \frac{\partial f}{\partial x^i}(p + t(x-p))\mathrm{d}t$, which are C^∞ functions on U, and eq. (2.41) becomes

$$f(x) - f(p) = \sum_i (x^i - p^i) g_i(x).$$

Additionally, we have

$$g_i(p) = \int_0^1 \frac{\partial f}{\partial x^i}(p + t(p-p))\mathrm{d}t = \int_0^1 \frac{\partial f}{\partial x^i}(p)\mathrm{d}t = \frac{\partial f}{\partial x^i}(p),$$

which completes the proof. $\square$

In the one-dimensional case the lemma says $f(x) = f(p) + (x-p)g_{(1)}(x)$, where $g_{(1)}(x)$ is some C^∞ function. Applying the lemma repeatedly to the function $g_{(1)}(x)$, one obtains higher-order terms

$$\begin{aligned} g_{(1)}(x) &= g_{(1)}(p) + (x-p)g_{(2)}(x) \\ g_{(k)}(x) &= g_{(k)}(p) + (x-p)g_{(k+1)}(x), \end{aligned}$$

where the $g_{(k)}$ are C^∞ functions. Inserting the higher-order terms with the shorthand $\Delta := x - p$

$$\begin{aligned} f(x) &= f(p) + \Delta\left[g_{(1)}(p) + \Delta g_{(2)}(x)\right] \\ &= f(p) + \Delta g_{(1)}(p) + (\Delta)^2\left[g_{(2)}(p) + \Delta g_{(3)}(x)\right] \\ &\ \ \vdots \\ &= f(p) + \Delta g_{(1)}(p) + (\Delta)^2 g_{(2)}(p) + \cdots + (\Delta)^k g_{(k)}(p) + (\Delta)^{k+1} g_{(k+1)}(x), \end{aligned} \tag{2.42}$$

where the last term is called the remainder and we denote the k-th power by $(\Delta)^k$ to distinguish from the components of a point. Let us investigate the functions $g_{(k)}$ at the point p. The case $k = 1$ follows trivially from the lemma as $g_{(1)}(p) = \frac{\mathrm{d}f}{\mathrm{d}x}(p)$. Taking $k = 2$ in eq. (2.42), differentiating twice and evaluating at p yields

$$\begin{aligned} \left.\frac{\mathrm{d}^2 f(x)}{\mathrm{d}(x)^2}\right|_{x=p} &= \frac{\mathrm{d}^2}{\mathrm{d}(x)^2}\left(f(p) + \Delta g_{(1)}(p) + (\Delta)^2 g_{(2)}(p) + (\Delta)^3 g_{(3)}(x)\right)\Big|_{x=p} \\ &= 2g_{(2)}(p) + \frac{\mathrm{d}^2}{\mathrm{d}(x)^2}\left((x-p)^3 g_{(3)}(x)\right)\Big|_{x=p} \\ &= 2g_{(2)}(p) + \left(3\cdot 2(x-p)g_{(3)}(x)\right)\Big|_{x=p} + \left.\left((x-p)^3\frac{\mathrm{d}^2 g_{(3)}(x)}{\mathrm{d}(x)^2}\right)\right|_{x=p} \\ &= 2g_{(2)}(p), \end{aligned}$$

hence $g_{(2)}(p) = \frac{1}{2}\frac{\mathrm{d}^2 f}{\mathrm{d}(x)^2}(p)$. In general the remainder, i.e. the $k+1$-th term of eq. (2.42), differentiated k times and evaluated at p is zero. The terms $(\Delta)^i g_{(i)}(p)$, $i < k$ differentiated k times are zero as well. Thus, differentiating eq. (2.42) k times and evaluating at p yields

$$\left.\frac{\mathrm{d}^k f}{\mathrm{d}(x)^k}\right|_{x=p} = \left.\frac{\mathrm{d}^k}{\mathrm{d}(x)^k}\left((\Delta)^k g_{(k)}(p)\right)\right|_{x=p} = \left.\frac{\mathrm{d}^k}{\mathrm{d}(x)^k}\left((x-p)^k g_{(k)}(p)\right)\right|_{x=p} = k!\,g_{(k)}(p).$$

We obtain that

$$g_{(m)}(p) = \frac{1}{m!}\frac{\mathrm{d}^m f}{\mathrm{d}(x)^m}(p), \quad m = 1, \dots, k,$$

which shows that the polynomial expansion eq. (2.42) agrees up to the last term with the Taylor series of f at p.

Note that if f is a C^∞ function defined on an open set U (not necessarily convex), then there is an $\varepsilon > 0$ such that $p \in B(p,\varepsilon) \subset U$, with $B(p,\varepsilon) = \{x \in \mathbb{R}^n \mid \|x-p\| < \varepsilon\}$ the open ball of radius ε. We can restrict the domain of f to $B(p,\varepsilon)$, which is convex, and the lemma applies.

Theorem 2.1. *Let $p \in \mathbb{R}^n$. The map $\phi : v_p \mapsto D_v|_p$ is an isormorphism from $T_p(\mathbb{R}^n)$ to $\mathfrak{D}_p(\mathbb{R}^n)$.*

Proof. To show that the map ϕ is an isormorphism, we need to check that it is one-to-one (injective) and onto (surjective). We first note that ϕ is linear, as can be seen easily from eq. (2.40). Furthermore, we know from linear algebra that a linear map is injective iff its kernel contains only the zero element. Hence we consider a tangent vector v_p with the property that $D_v|_p$ is the zero derivation. We will show that it follows that v_p itself must be the zero vector. Expressing v_p in the standard basis as $v_p = \sum_i v^i e_i$ and taking f to be the j-th coordinate function x^j, we obtain by inserting into the definition of the directional derivative eq. (2.38)

$$0 = D_v f = D_v x^j = \sum_i v^i \frac{\partial x^j}{\partial x^i}.$$

Clearly $\frac{\partial x^j}{\partial x^i} = 0$ except when $i = j$ where it is 1. Therefore we have $0 = v^j$ and since this holds for all j it follows that $v_p = 0$.

To prove surjectivity, let w be an arbitrary derivation at p. Similar to the previous paragraph, we define a vector $v_p = \sum_i v^i e_i$, with the numbers v^i given by $v^i = w(x^i)$. We will now show that $w = D_v|_p$. To that end, consider any C^∞ function f in a convex neighborhood around p. By lemma 2.6, f can be written as

$$f(x) = f(p) + \sum_i (x^i - p^i) g_i(x), \quad g_i(p) = \frac{\partial f}{\partial x^i}(p).$$

Now we apply the derivation w to both sides

$$wf = w\left(f(p)\right) + \sum_i w\left((x^i - p^i) g_i(x)\right),$$

and note that $w\left(f(p)\right)$ and $w(p^i)$ are both zero by lemma 2.5. Using the product rule, we obtain

$$\begin{aligned} wf &= \sum_i w\left((x^i - p^i)g_i(x)\right) \\ &= \sum_i \left(w(x^i) - w(p^i)\right) g_i(p) + \sum_i (p^i - p^i) w g_i(x) \\ &= \sum_i w(x^i) \frac{\partial f}{\partial x^i}(p) \\ &= \sum_i v^i \frac{\partial f}{\partial x^i}(p). \end{aligned}$$

Since this holds for any f and w was arbitrary, we conclude that $w = D_v|_p$. □

This theorem shows that a tangent vector $v_p \in T_p\mathbb{R}^n$ *is* a derivation. Hence, the notation $v_p f$ means to take the derivative in direction v_p of the smooth function f as suggested by eq. (2.40). In particular, we can consider the standard basis $\{e_i\}$ for the tangent space $T_p\mathbb{R}^n$ to obtain the following useful result:

Corollary 2.1. For any $p \in \mathbb{R}^n$, the n derivations

$$\left.\frac{\partial}{\partial x^1}\right|_p, \ldots, \left.\frac{\partial}{\partial x^n}\right|_p \text{ defined by } \left.\frac{\partial}{\partial x^i}\right|_p f = \frac{\partial f}{\partial x^i}(p)$$

form a basis for $T_p\mathbb{R}^n$, which therefore is a n-dimensional vector space.

Proof. Apply the previous theorem and note that $\left.\frac{\partial}{\partial x^i}\right|_p = D_{e_i}|_p$. □

We will frequently make use of this corollary in the following way: Consider a coordinate system on $\mathbb{R}^n$ given by some coordinate mapping φ. We can construct a basis for the tangent space at any point $p \in \mathbb{R}^n$ by taking partial derivatives of φ. This approach to tangent vectors might seem more complicated and abstract, when compared to the intuitive geometric approach where a tangent vector at p is visualized as an arrow at $\varphi(p)$. However, consider a *different* coordinate mapping ψ, the tangent vector at p will be a different arrow at $\psi(p)$, and one would need to figure out a way to identify arrows at $\varphi(p)$ with arrows at $\psi(p)$. Viewing tangent vectors as derivations at p is the most intrinsic way, since it is independent from the particular coordinate system. This approach to differential geometry is hence also called *coordinate-free*. Of course, since the two approaches are equivalent, in practice it is useful and sensible to keep the intuitive "arrow" point of view in mind.

2.3.2.1 Tensor Notation

Most of the technical machinery of modern differential geometry is built up using tensors, hence we have to introduce a few notation conventions. In an informal way, tensors can be seen as an extension of linear algebra to multidimensional arrays. For now we can think of a tensor as a collection of numbers indexed in some systematic way, similar in spirit to a vector or a matrix. In contrast to linear algebra, where the focus is on the algebraic relation between objects[6], tensor calculus puts the focus on the individual elements, called *components*, of the tensor. To this end, an indicial notation is used. Tensor calculus uses both lower and upper indices, i.e. upper indices denote components, not exponentiation. The number of different indices is called the *order* (or degree, rank) of the tensor. Usually the range of the indices is clear from the context and not given explicitly. Whereas the name of the index does not matter (widely used are the Latin letters $i, j, k \dots$ and the Greek letters α, β, μ, ν), their vertical (upper vs. lower) as well as horizontal (first, second, third...) position is of significance. To avoid ambiguities, we may use an additional dot to clarify the order of index slots in a tensor with mixed upper and lower indices. For example, the rank 3 tensor $A^{k\cdot j}_{\cdot i \cdot}$ has k as the first index, i as second and j as third.

One of the most important notation conventions is the so-called Einstein summation convention, popularized by Einstein and his theory of general relativity.

Definition 2.14 (Einstein Notation)**.** If any monomial term contains the same index twice, once as a lower and once as an upper index, summation over that index is implied. The index that is summed over is also called *dummy index*, and the summation over a dummy index is known as *contraction*. In many cases the dummy index of a contraction can be renamed freely, e.g. $x_j y^j$ and $x_i y^i$ mean the same thing.

For example, matrix-vector multiplication can be expressed with the help of the summation convention as follows:

$$A = \begin{bmatrix} A_{11} & \dots & A_{1j} \\ \vdots & & \vdots \\ A_{i1} & \dots & A_{ij} \end{bmatrix}, \; x = \begin{bmatrix} x^1 \\ \vdots \\ x^j \end{bmatrix}$$

$$Ax = A_{ij} x^j \left(= \sum_j A_{ij} x^j \right).$$

One of the advantages of indicial tensor notation is that expressions are given in terms of the individual tensor components, which are just numbers. If one wants to do calculus this is a good thing, and sometimes a problem can be solved more easily. Consider the following example of quadratic form minimization in linear algebra notation

$$\min_x \left\{ f(x) = \tfrac{1}{2} x^T Q x + b^T x \right\}, \quad Q \in \mathbb{R}^{n \times n}, \; x, b \in \mathbb{R}^n. \tag{2.43}$$

[6]In the matrix equation $Ax = b$ and its solution $x = A^{-1}b$, the elements of the matrix A and vectors x, b play a secondary role.

Using tensor notation, eq. (2.43) reads

$$\min_x \left\{ f(x) = \tfrac{1}{2} Q_{ij} x^i x^j + b_i x^i \right\}. \tag{2.44}$$

Minimizing this quadratic form involves computing the gradient of f with respect to x. Computing the gradient from the linear algebra notation eq. (2.43) turns out to be quite involved if one does not *know* the result a priori (for example, by looking it up in the matrix cookbook which contains the identity $\frac{\partial x^T B x}{\partial x} = (B + B^T)x$ without further explanation). The reason for this is that linear algebra notation hides the components of matrices and vectors, but computing the gradient needs access to these components. In tensor notation on the other hand, the task is simple: Just apply the product rule to eq. (2.44) to find

$$\frac{\partial f(x)}{\partial x^k} = \frac{1}{2} Q_{ij} \frac{\partial x^i}{\partial x^k} x^j + \frac{1}{2} Q_{ij} x^i \frac{\partial x^j}{\partial x^k} + b_i \frac{\partial x^i}{\partial x^k}. \tag{2.45}$$

The quantity $\frac{\partial x^i}{\partial x^k}$ obviously is zero whenever $i \neq k$ and 1 if $i = k$. In tensor calculus, there exists a special symbol for this situation.

Definition 2.15. The Kronecker delta symbol is defined as

$$\delta^i_j = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}.$$

The Kronecker delta typically appears in a contraction and has the effect of “renaming an index”, since it is easy to see that $\delta^i_j b_i = b_j$. Hence, eq. (2.45) simplifies to

$$\begin{aligned} \frac{\partial f(x)}{\partial x^k} &= \frac{1}{2} Q_{ij} \delta^i_k x^j + \frac{1}{2} Q_{ij} x^i \delta^j_k + b_i \delta^i_k \\ &= \frac{1}{2} Q_{kj} x^j + \frac{1}{2} Q_{ik} x^i + b_k \\ &= \frac{1}{2} Q_{ki} x^i + \frac{1}{2} Q_{ik} x^i + b_k, \end{aligned} \tag{2.46}$$

where in the last line we renamed the dummy index of the first term $j \to i$. Thanks to the rules of the Einstein convention, there is no ambiguity how to interpret e.g. the term $Q_{ij} x^i \delta^j_k$. Q and δ both contain the index j, as a lower and upper index respectively. Thus summation is invoked and the Kronecker delta is absorbed into Q, renaming the index $j \to k$.

We see that $Q_{ki} x^i$ corresponds in linear algebra notation to the matrix-vector multiplication Qx, whereas $Q_{ik} x^i$ corresponds to $Q^T x$. Therefore we can state that the expression for the gradient of (2.43) is given by $\nabla f = \frac{1}{2}\left(Q + Q^T\right) x + b$. It is worth noting that the tensor notation reveals additional information about the quadratic form in a very natural way. The combination of indices in eq. (2.46) tells that the range of the first and second index of Q must be the same. In other words, the matrix Q must be quadratic. Of course we already knew that, since we started with a quadratic form in the beginning. Nonetheless it is nice that this fact is supported by the notation as well. Moreover, we can see that if $Q_{ki} = Q_{ik}$, i.e. the matrix is symmetric, the gradient simplifies to $\frac{\partial f(x)}{\partial x^k} = Q_{ki}x^i + b_k$, which in linear algebra notation corresponds to $\nabla f = Qx + b$.

We conclude this section by applying tensor notation to the concepts that we have already developed. The situation to sum over an index occurs ubiquitously in differential geometry, which is precisely the reason why the Einstein convention has been invented. In particular, we can express the coordinate representation of a vector v with respect to some basis $\{b_i\}$ as $v = v^i b_i$. Note that in this expression we think of v as a geometric vector, that is, an object independent of the coordinate system. The *numbers* v^i are its representation once a particular coordinate system has been chosen. We may use the boldface notation in case we wish to emphasize the geometric nature of a vector.

Likewise, we can now denote the directional derivative (see eq. (2.38)) of a function f at a point p very concisely as

$$D_v f|_p = v^i \frac{\partial f}{\partial x^i}(p).$$

In this expression the summation convention applies, i.e. the right hand side is assumed to be summed over i. This is consistent with the indexing convention, since we regard an "upper index in the denominator" as a lower index. Similarly, by applying corollary 2.1 any tangent vector $v \in T_p(\mathbb{R}^n)$ can be written as

$$v = v^i \frac{\partial}{\partial x^i}\bigg|_p, \tag{2.47}$$

where $\frac{\partial}{\partial x^i}$ are the partial derivatives of the coordinate mapping at p.

2.3.2.2 Coordinate Bases

We now bring the rather abstract concepts about tangent vectors developed in section 2.3.2 to more concrete geometric applications. Consider the position vector $\mathbf{R}$, which represents points in Euclidean space with respect to an arbitrarily chosen origin, and introduce a coordinate system φ given by $x = (x^i)$. Similar to eq. (2.36) we think about the coordinate system as a set of *coordinate functions* x^i. In particular, since the coordinate mapping is a homeomorphism, we have that the coordinates are functions of the position vector $x = x(\mathbf{R})$, and vice-versa the position vector is a function of the coordinates $\mathbf{R} = \mathbf{R}(x)$. For example, consider the two dimensional Euclidean plane $\mathbb{E}^2$ referred to the natural Cartesian coordinate system. This means that we have the coordinates $x^1 = x$, $x^2 = y$, i.e. $\mathbf{R} = \mathbf{R}(x, y)$. If we refer $\mathbb{E}^2$ to a polar coordinate system, we have $x^1 = r$ (the radius), and $x^2 = \theta$ (the angle) and consequently $\mathbf{R} = \mathbf{R}(r, \theta)$. To any combination of (x, y) resp. (r, θ) coordinates there corresponds a specific position vector $\mathbf{R}$.

By the fundamental result of corollary 2.1, we obtain a basis for the tangent space by partial differentiation of the coordinate mapping. Thus we define the *covariant basis vectors* $\mathbf{X}_i$ as

$$\mathbf{X}_i = \frac{\partial \mathbf{R}(x)}{\partial x^i}. \tag{2.48}$$

The term *covariant* refers to the way the basis vectors transform under a change of coordinates. As customary in differential geometry, covariance is indicated by lower indices. Note that $\mathbf{X}_i$ is not the i-th component of a tuple of numbers, it rather denotes the i-th basis vector. Previously we made a point to not think of a geometric vector as a tuple of numbers, hence whenever an index appears with a geometric vector, indicated by boldface, it cannot refer to components. Unfortunately, the conflation between vectors and tuples of numbers (see section 2.3.1.1) makes it difficult to keep track of which object means what. To make matters worse, notation is often guided by conventions.[7]

Example 2.1. Refer the Euclidean plane $\mathbb{E}^2$ to a polar coordinate system, i.e. $x^1 = r,\ x^2 = \theta$. The polar coordinate system is set up as follows: Choose a designated point to be the origin, further select an arbitrary axis to be the axis corresponding to $\theta = 0$. For convenience we will let the horizontal axis correspond to $\theta = 0$. It is important to specify the (any) coordinate system in geometric terms, without reference to a "background" Cartesian coordinate system. By that we mean to refrain from definitions like *let the x-axis correspond to $\theta = 0$*. Euclidean space does not have an *x-axis* built into it.

We wish to compute the covariant basis at the points with coordinates $\left(3, \frac{\pi}{4}\right)$ and $\left(1, \frac{5\pi}{6}\right)$, see fig. 2.13.

Since we do not have an explicit expression for the coordinate mapping, we adopt a geometric approach in the spirit of Euclid. To find the basis vector $\mathbf{X}_r$, compute the partial derivative of $\mathbf{R}(r,\theta)$ w.r.t. r by constructing the limit

$$\mathbf{X}_r = \frac{\partial \mathbf{R}(r,\theta)}{\partial r} = \lim_{h\to 0} \frac{\mathbf{R}(r+h,\theta) - \mathbf{R}(r,\theta)}{h}.$$

It is easy to see that this vector points in radial direction and has unit length. To find the basis vector $\mathbf{X}_\theta$, compute in a similar fashion

$$\mathbf{X}_\theta = \lim_{h\to 0} \frac{\mathbf{R}(r,\theta+h) - \mathbf{R}(r,\theta)}{h}.$$

This vector points in tangential direction to the circle of radius r and its length is proportional to r. This additional factor r comes from the fact that the "speed" of the curve representing the circle is proportional to r – larger circles need to be traveled "faster" as θ goes from 0 to 2π.

[7] *The old joke that "differential geometry is the study of properties that are invariant under change of notation" is funny primarily because it is alarmingly close to the truth. Every geometer has his or her favorite system of notation, and while the systems are all in some sense formally isomorphic, the transformations required to get from one to another are often not at all obvious to students.* – [Lee, 2012]

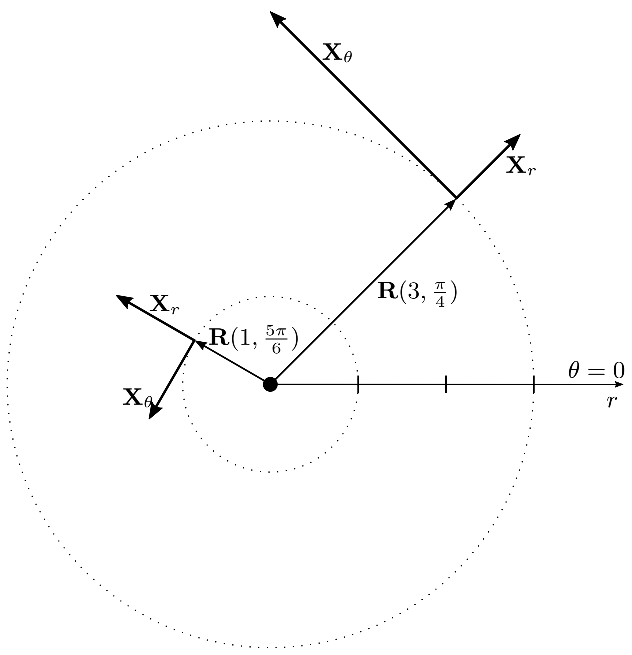


Figure 2.13: The position vectors representing the points with coordinates $\left(3, \frac{\pi}{4}\right)$ and $\left(1, \frac{5\pi}{6}\right)$ in a polar coordinate system along with the basis vectors $\mathbf{X}_r, \mathbf{X}_\theta$.

The covariant basis can be used to represent other vectors: Any vector $\mathbf{V}$ is given as the unique linear combination $\mathbf{V} = v^i \mathbf{X}_i$. The numbers v^i are called *contravariant components* of $\mathbf{V}$ w.r.t. to the coordinate basis, or simply coordinates of $\mathbf{V}$. The term contravariant refers to the way v^i transforms under a change of coordinates – this will be explained in section 2.3.4, where we introduce the tensor property. Contravariance is indicated by upper indices, and we note again that in the expression $\mathbf{V} = v^i \mathbf{X}_i$, the v^i are *numbers* whereas the $\mathbf{X}_i$ are *vectors*. We also emphasize that the covariant basis might vary from point to point, see fig. 2.13.

Equivalently we can say that a vector $v \in T_p\mathbb{R}^n$ is represented in a coordinate system (x^i) by $v = v^i \frac{\partial}{\partial x^i}$. The equivalence comes of course from the fact that we defined the position vector $\mathbf{R}$ as representing points in space. The crucial point here is that geometric vectors and points in space are in a sense fixed, "physical" objects, while their *representations* are just tuples of numbers that depend on a coordinate system.

A Cartesian coordinate system in the two-dimensional plane is characterized by the fact that the coordinate axes are orthogonal. A simple calculation in the spirit of example 2.1 shows that in this case the covariant basis vectors are *constant* at all points. Hence we consider Euclidean space with a Cartesian coordinate system as a special case, in which by coincidence two identical arrows at different locations happen to have the same coordinate representations. It is the silent assumption of this special case, that allows us to routinely think about vectors as objects having length and direction, discarding their location. In particular, we emphasize that in general the coordinate representations of two vectors at different locations cannot be e.g. added. The coordinates gain meaning only in conjunction with the basis vectors, which might vary from point to point. This means that the tangent spaces $T_p\mathbb{R}^n$ and $T_q\mathbb{R}^n$ at two points $p, q \in \mathbb{R}^n$, $p \neq q$, are really two separate copies of $\mathbb{R}^n$. In general the coordinate representation of a tangent vector in one space cannot be transferred directly to the other. Only in Euclidean space with a Cartesian coordinate system we can informally take the coordinate representation to *be* the vector, and adding coordinate representations of vectors at different points works correctly.

2.3.2.3 Change of Coordinates

We now investigate the relation between different coordinate systems. Due to the central role of tangent vectors established in section 2.3.2, we are interested in the way a change of coordinates affects tangent vectors at a point. To that end, we first explore how a smooth mapping $F : \mathbb{R}^n \to \mathbb{R}^m$ acts on elements of the tangent space. We consider a linear map between the tangent spaces

$$\mathrm{d}F_p : T_p\mathbb{R}^n \to T_{F(p)}\mathbb{R}^m, \tag{2.49}$$

called *differential* of F at p. Let $v_p \in T_p\mathbb{R}^n$, then $\mathrm{d}F_p(v_p)$ is the tangent vector in $T_{F(p)}\mathbb{R}^m$ defined through its action on a function $f \in C^\infty(\mathbb{R}^m)$

$$(\mathrm{d}F_p(v_p))\, f = v_p(f \circ F). \tag{2.50}$$

Since the composition $f \circ F \in C^\infty(\mathbb{R}^n)$, applying the derivation $v_p \in T_p\mathbb{R}^n$ makes sense.

Let $(x^1, \ldots, x^n)$ be the coordinates on $\mathbb{R}^n$ and $(y^1, \ldots, y^m)$ the coordinates on $\mathbb{R}^m$ respectively. We have seen that a basis for the tangent space $T_p\mathbb{R}^n$ is given by $\left\{\mathbf{X}_i := \frac{\partial}{\partial x^i}\big|_p\right\}$, and similarly $\left\{\mathbf{Y}_j := \frac{\partial}{\partial y^j}\big|_{F(p)}\right\}$ is a basis for the tangent space $T_{F(p)}\mathbb{R}^m$. As before, it is important to remember that indices are used for two different meanings here: The object x^i is a number, but $\mathbf{X}_i$ is a vector, indicated by boldface notation. The action of the linear map $\mathrm{d}F_p$ on a basis vector is given by

$$(\mathrm{d}F_p\,(\mathbf{X}_i))\, f = \mathbf{X}_i(f \circ F).$$

Applying the chain rule, we find

$$\mathbf{X}_i(f \circ F) = \frac{\partial f}{\partial y^j}\,(F(p))\,\frac{\partial F^j}{\partial x^i}(p) = \left(\mathbf{Y}_j \frac{\partial F^j}{\partial x^i}(p)\right) f,$$

where F^j is the j-th component function of F, and consequently

$$\mathrm{d}F_p\,(\mathbf{X}_i) = \frac{\partial F^j}{\partial x^i}(p)\mathbf{Y}_j. \tag{2.51}$$

Note that the Einstein convention allows a very compact notation of the summation involved in the chain rule. Applying eq. (2.51) to a tangent vector $v \in T_p\mathbb{R}^n$ yields

$$v = v^i \mathrm{d}F_p\left(\mathbf{X}_i\right) = v^i \frac{\partial F^j}{\partial x^i}(p)\mathbf{Y}_j.$$

Hence the matrix of $\mathrm{d}F_p$ relative to the bases $\mathbf{X}_i$ and $\mathbf{Y}_j$ is given by

$$\begin{bmatrix} \frac{\partial F^1}{\partial x^1}(p) & \cdots & \frac{\partial F^1}{\partial x^n}(p) \\ \vdots & \ddots & \vdots \\ \frac{\partial F^m}{\partial x^1}(p) & \cdots & \frac{\partial F^m}{\partial x^n}(p) \end{bmatrix}, \tag{2.52}$$

which is precisely the *Jacobian matrix* of F at p. The Jacobian matrix is also called *pushforward*, since it “pushes” tangent vectors from the domain of F to the codomain.

The Jacobian can also be expressed in coordinates, where we denote by φ the coordinate mapping on the domain $\mathbb{R}^n$ and by ψ the coordinate mapping on the codomain $\mathbb{R}^m$. A coordinate expression for the point p is thus given by $\hat{p} = \varphi(p)$, and the mapping F in coordinates reads $\hat{F} = \psi \circ F \circ \varphi^{-1} : \varphi\left(\mathbb{R}^n\right) \to \psi(\mathbb{R}^m)$. Then a short computation yields

$$\mathrm{d}F_p\left(\mathbf{X}_i\right) = \frac{\partial \hat{F}^j}{\partial x^i}(\hat{p})\mathbf{Y}_j, \tag{2.53}$$

which says that the coordinate expression of the Jacobian is simply eq. (2.51), where the partial derivative of F at p is replaced with the respective coordinate representation.

We can use these results as follows: Given two coordinate systems φ and ψ on $\mathbb{R}^n$ with coordinate functions $\left(x^i\right), \left(x^{i'}\right)$ respectively, the tangent vector at a point p can be expressed either in the basis $\{\mathbf{X}_i\}$ or in the basis $\{\mathbf{X}_{i'}\}$. The two coordinate systems are often called the unprimed or “old” and primed or “new” coordinate system respectively.[8] We wish to know how the two representations of a tangent vector are related. To that end, we let $F = \mathrm{id}$ be the identity mapping and apply eq. (2.53). We see that in this case we get a mapping $\hat{F} = \psi \circ \varphi^{-1}$ known as *transition map* between the coordinate systems. Thanks to the coordinate mappings being homeomorphisms, this is a smooth map. An expression for the transition map is given by

$$\psi \circ \varphi^{-1}(x) = x'(x) = \left(x^{i'}(x)\right),$$

where we follow the common practice in differential geometry of not distinguishing between the coordinates as functions and the coordinates as vector representation, i.e. tuples of numbers. In fact, the choice of the variable x for the coordinate functions is meant to encourage this casual conflation, where we sometimes think of x as a function and sometimes as coordinate vector. For instance, in the expression $x^{i'}(x)$ we think of $x^{i'}$ as the i-th coordinate function in the primed coordinate system, but of x as the coordinate vector of p in the unprimed coordinate system. Usually it is clear from the context which

[8]Note that we placed the prime at the index.

interpretation applies. Inserting into eq. (2.53) we get

$$\mathbf{X}_i = \frac{\partial}{\partial x^i}\bigg|_p = \frac{\partial x^{i'}}{\partial x^i}(\hat{p})\mathbf{X}_{i'} = \frac{\partial x^{i'}}{\partial x^i}(\varphi(p))\,\frac{\partial}{\partial x^{i'}}\bigg|_p, \tag{2.54}$$

where we have back-substituted the shorthands $\mathbf{X}_i := \frac{\partial}{\partial x^i}\big|_p$ in order to make the connections clear. Now we apply this expression to the representation of a tangent vector $v = v^i\mathbf{X}_i = v^{i'}\mathbf{X}_{i'} = v^i\frac{\partial x^{i'}}{\partial x^i}(\hat{p})\mathbf{X}_{i'}$, from which we get that the components of a tangent vector transform according to

$$v^{i'} = \frac{\partial x^{i'}}{\partial x^i}(\hat{p})v^i, \tag{2.55}$$

and hence the Jacobian of the coordinate transition map is

$$\begin{bmatrix} \frac{\partial x^{1'}}{\partial x^1}(\hat{p}) & \cdots & \frac{\partial x^{1'}}{\partial x^n}(\hat{p}) \\ \vdots & \ddots & \vdots \\ \frac{\partial x^{n'}}{\partial x^1}(\hat{p}) & \cdots & \frac{\partial x^{n'}}{\partial x^n}(\hat{p}) \end{bmatrix}.$$

This Jacobian, which transforms the component representation of a tangent vector from the unprimed to the primed coordinate system, will be denoted $J_i^{i'} = \frac{\partial x^{i'}}{\partial x^i}$. It allows to express the transformation rule very concisely as $v^{i'} = J_i^{i'}v^i$. Note that we have to evaluate the Jacobian at the coordinate representation of p in the unprimed coordinate system. This is easy to remember: Use the representation of p in that coordinate system in which the vector we want to transform "lives". In this case we transform the vector v^i, which lives in the unprimed system.

Example 2.2. Refer the two-dimensional Euclidean plane to a polar coordinate system φ with coordinate functions (r, θ) and to the standard Cartesian coordinate system ψ with coordinate functions (x, y). Let p be the point in $\mathbb{R}^2$ whose polar coordinates are $(2, \pi)$ and let $v \in T_p\mathbb{R}^2$ be the tangent vector whose polar coordinates are $(v^i) = (-3, -1)$. We are going to find the Cartesian coordinates of v.

The transition map between polar and Cartesian coordinates is given by $(x, y) = (r\cos\theta, r\sin\theta)$, where we restrict (r, θ) to a suitable subset of the plane. We compute

$$J_i^{i'} = \begin{bmatrix} \frac{\partial r\cos\theta}{\partial r} & \frac{\partial r\cos\theta}{\partial\theta} \\ \frac{\partial r\sin\theta}{\partial r} & \frac{\partial r\sin\theta}{\partial\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & -r\sin\theta \\ \sin\theta & r\cos\theta \end{bmatrix}.$$

Now $v = -3\frac{\partial}{\partial r}\big|_p - \frac{\partial}{\partial\theta}\big|_p$ in polar coordinates, and we obtain

$$v' = J_i^{i'}\big|_{(r,\theta)=(2,\pi)}v^i = \begin{bmatrix} -1 & 0 \\ 0 & -2 \end{bmatrix}\begin{bmatrix} -3 \\ -1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}.$$

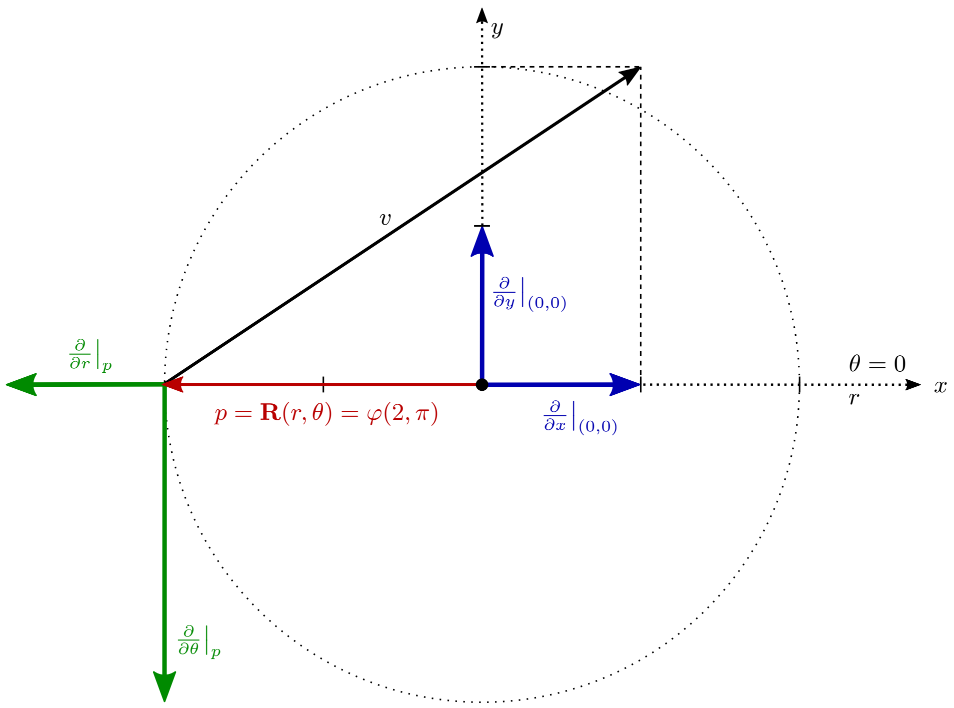

y
v
$\frac{\partial}{\partial y}\big|_{(0,0)}$
$\frac{\partial}{\partial r}\big|_p$
$\theta = 0$
x
r
$p = \mathbf{R}(r,\theta) = \varphi(2,\pi)$
$\frac{\partial}{\partial x}\big|_{(0,0)}$
$\frac{\partial}{\partial \theta}\big|_p$

Figure 2.14: The tangent vector v at the point p with polar coordinates $(2,\pi)$ can either be expressed as $v = -3\frac{\partial}{\partial r}\big|_p - \frac{\partial}{\partial \theta}\big|_p$ in polar coordinates or as $v = 3\frac{\partial}{\partial x}\big|_p + 2\frac{\partial}{\partial y}\big|_p$ in Cartesian coordinates. Note that the Cartesian basis at p is just the canonical basis (blue) shifted to p. Illustration of the Cartesian basis at p would overlap with the red position vector.

Hence in Cartesian coordinates the tangent vector is given by $v = 3\frac{\partial}{\partial x}\big|_p + 2\frac{\partial}{\partial y}\big|_p$, see fig. 2.14.

2.3.3 Topological Manifolds

One might wonder why such a sophisticated framework is needed, if the task is simply to convert from polar to Cartesian coordinates. The reason for this is twofold. First, the theory carries seamlessly to arbitrary manifolds. This allows to apply the concepts not only in flat spaces like $\mathbb{R}^n$, but also for instance on a sphere, a curved surface embedded in $\mathbb{R}^3$ etc. Second, in conjunction with tensors this approach provides a principled way to do calculations in coordinates. We have stated that coordinate systems are necessary for solving geometric problems by means of calculus. This led to the problem that expressions in coordinates quickly became complicated. In particular, the result of a calculation inevitably carries artifacts of the coordinate system. Recall the differential equation for the catenoid in Cartesian coordinates $f_{xx}(1+f_y^2)+f_{yy}(1+f_x^2)-2f_{xy}f_xf_y = 0$. This equation has a geometric meaning, but it also exhibits artifacts that are due to the Cartesian coordinate system. It is difficult to disentangle these properties. Tensors provide a principled way to formulate expressions where artifacts of the coordinate system are present in a *systematic* way.

Before we turn to geometrically invariant objects that can be obtained through tensor calculus, let us give a definition of manifolds. A manifold is a space that looks locally like Euclidean space, i.e. it is flat. More formally, we call the space M topological manifold of dimension n if

- M is a Hausdorff space.
- M is second-countable.
- M is locally Euclidean of dimension n.

While the first two conditions are technical, the third one is the most important for practical purposes. It means that any point $p \in M$ has a neighborhood that is homeomorphic to $\mathbb{R}^n$. This is expressed through the notion of *coordinate charts*. If M is a topological n-manifold, a chart on M is given by the pair (U, φ), where $U \subset M$ and $\varphi : U \to \hat{U} \subset \mathbb{R}^n$ is a homeomorphism from U to a subset of $\mathbb{R}^n$. Clearly the mapping φ is conceptually the same as the coordinate mapping between Euclidean space and $\mathbb{R}^n$ that we introduced in eq. (2.36). The difference is that now it does not map the whole space, but a subset of M into a subset of $\mathbb{R}^n$, see fig. 2.15. Therefore the mapping φ is often called *local* coordinate chart, with the corresponding *local coordinates*. With some care[9], we can as-

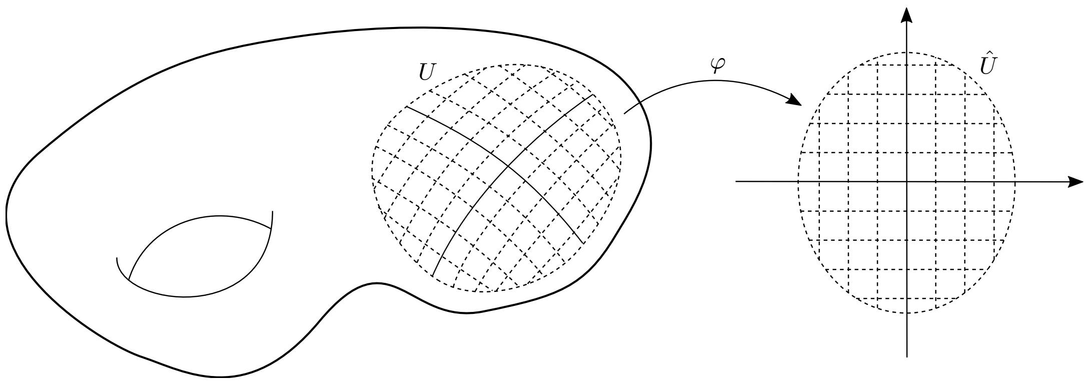


Figure 2.15: A coordinate chart mapping a subset $U \subset M$ into a subset $\hat{U} \subset \mathbb{R}^n$.

semble a collection of charts whose domains cover all of M. The collection of these charts is called the *atlas* $\mathcal{A}$ for M, and if any two charts in $\mathcal{A}$ are diffeomorphic it is a smooth atlas. If additionally $\mathcal{A}$ is *maximal*, meaning it is not contained in any larger atlas, we say that $\mathcal{A}$ is a *smooth strucure on M*. It is this smooth structure that allows to talk about differentiable functions on M. For example, consider the Euclidean space $\mathbb{R}^n$ with the single chart $(\mathbb{R}^n, \mathrm{id})$, where id is the identity mapping. Clearly this chart is smooth, and the atlas consisting of this chart is maximal. We call this atlas the *standard smooth*

[9]We need to make sure that for two overlapping charts (U, φ) and (V, ψ), $U \cap V \neq \emptyset$ the transition map $\psi \circ \varphi^{-1}$ is a diffeomorphism.

structure on $\mathbb{R}^n$, and it makes $\mathbb{R}^n$ into a smooth n-manifold. It is important to realize that even though we almost instinctively imagine a Cartesian coordinate system with x, y, z axes when we think about $\mathbb{R}^3$, a manifold does not come with a predetermined coordinate system. In particular, certainly the manifold $\mathbb{R}^3$ does not come with Cartesian coordinates. One might equally well work with e.g. spherical coordinates. However, we will refer to Cartesian coordinates on $\mathbb{R}^n$ as *standard* coordinates.

For general manifolds M the situation in a sense is less ambiguous, because usually it is not possible to cover M with a single chart. One inherently has different charts and hence different local coordinates. In this case, the temptation to speak about (or assume) *"the"* coordinate system does not exist.

2.3.4 The Tensor Property

In section 2.3.2.2 we have seen that the Jacobian $J_i^{i'}$ transforms a tangent vector at p from the unprimed to the primed coordinate system. Thanks to the framework of manifolds, the transition map between coordinate systems is a diffeomorphism, meaning the Jacobian is invertible. It is natural to define the inverse Jacobian $J_{i'}^{i} = \frac{\partial x^i}{\partial x^{i'}}$, which transforms from the primed to the unprimed coordinate system, where we have to remember to evaluate $J_{i'}^{i}$ at the coordinate representation of p w.r.t. the primed coordinate system. The fact that the two Jacobians are inverses[10] of each other is expressed as

$$J_i^{i'} J_{j'}^{i} = \delta_{j'}^{i'} \quad \text{and} \quad J_{i'}^{i} J_j^{i'} = \delta_j^i. \tag{2.56}$$

Tensors are objects whose components transform in a certain way. In particular, T_i is called *covariant tensor*, if the components are related by

$$T_{i'} = T_i J_{i'}^{i}. \tag{2.57}$$

Likewise, if the components of T^i transform as

$$T^{i'} = T^i J_i^{i'}, \tag{2.58}$$

we call T^i a *contravariant tensor*. Similar relations can be given for higher order tensors. For example we call T_{ij} a covariant tensor of order two if it transforms as $T_{i'j'} = T_{ij} J_{i'}^{i} J_{j'}^{j}$. Co- and contravariant indices can be mixed, for example $T_{j'}^{i'} = T_j^i J_i^{i'} J_{j'}^{j}$ is a mixed tensor of order two. More formally, a rank (k, l) tensor has k contravariant (upper) and l covariant (lower) indices. Whereas in $\mathbb{R}^n$ we can informally think of rank 1 tensors as "vectors" and of rank 2 tensors as "matrices", it is advisable to see a tensor simply as a indexed collection of numbers that transform in a certain way.

[10]To see the inverse relationship, one has to take care to evaluate the Jacobians *in a common coordinate system*, which usually involves transforming the entries of one Jacobian into the coordinate system of the other.

Covariant means "transforms in the same way as the basis" and is indicated by lower indices. Similarly, the components of a contravariant tensor transform in the opposite way as the basis, indicated by upper indices. To demonstrate this we recall eq. (2.54), which says that a basis vector $\mathbf{X}_i$ transforms under a change of coordinates as $\mathbf{X}_i = \mathbf{X}_{i'} \frac{\partial x^{i'}}{\partial x^i}(\varphi(p))$. Applying this to a tangent vector v, we found in eq. (2.55) that its components transform as $v^{i'} = v^i \frac{\partial x^{i'}}{\partial x^i}(\varphi(p))$. Note that the transformation of the *unprimed* basis is the same as the transformation of the *primed* vector components. Due to the inverse relationship of the Jacobians, we find that the unprimed vector components transform as $v^i = v^{i'} \frac{\partial x^i}{\partial x^{i'}}(\psi(p))$, which is opposite (contravariant) to the unprimed basis vector.

It is important to note that not all indexed objects are tensors. For instance, the partial derivative $\frac{\partial T^i}{\partial x^j}$ of a tensor is not a tensor, and neither is the Jacobian $J^i_{i'}$. The components of an indexed object must transform as eq. (2.57) or eq. (2.58) to qualify as a tensor.

We are now in the position to justify that we called the basis vector in eq. (2.48) "covariant". We start with the expression for the basis vector in the primed coordinate system

$$\mathbf{X}_{i'} = \frac{\partial \mathbf{R}(x')}{\partial x^{i'}},$$

and then see how it relates to the unprimed system. The key observation is that the position vector $\mathbf{R}$ is a geometric vector – it exists independent from any coordinate system. In particular, we can express the same position vector as a function of the unprimed coordinates by substituting $x(x')$, i.e. $\mathbf{R}(x') = \mathbf{R}\left(x(x')\right)$. Applying the chain rule we get

$$\mathbf{X}_{i'} = \frac{\partial \mathbf{R}(x(x'))}{\partial x^{i'}} = \left.\frac{\partial \mathbf{R}(x)}{\partial x^i}\right|_{x=x(x')} \frac{\partial x^i}{\partial x^{i'}} = \mathbf{X}_i J^i_{i'}, \tag{2.59}$$

which shows that the basis indeed transforms according to eq. (2.57) and is thus a covariant tensor.

Next we introduce one of the most important objects in differential geometry: The metric tensor. To that end, we recall that the dot product for geometric vectors (definition 2.11) played an important role in establishing various sorts of geometric relations. In a vector space, the equivalent of the dot product is the Euclidean inner product (definition 2.13). This is even more important, since an inner product provides a metric to the vector space, which in turn can be used to *define* the length of vectors. It is no exaggeration to say that geometry begins with the measurement of length, hence the Euclidean inner product is indeed fundamental. With the tensor framework available, we ask the question how the dot product looks "in coordinates". Let $\mathbf{U} = u^i\mathbf{X}_i$ and $\mathbf{V} = v^i\mathbf{X}_i$ be two vectors located at the same point, the dot product is given by $\mathbf{U}\cdot\mathbf{V} = u^i\mathbf{X}_i\cdot v^j\mathbf{X}_j = u^i v^j\left(\mathbf{X}_i \cdot \mathbf{X}_j\right)$. The indicial tensor notation facilitates the reordering of terms in the last equality. The components u^i, v^i are just numbers, and in a multiplication of numbers we are free to shuffle terms to our liking. We just need to make sure that the dot product is computed between the two basis vectors. This pairwise dot product between the basis vectors is the metric tensor. We state this result as a definition.

Definition 2.16 (Metric tensor)**.** The *covariant metric tensor* g_{ij} is the pairwise dot product between the covariant basis vectors

$$g_{ij} = \mathbf{X}_i \cdot \mathbf{X}_j = \frac{\partial}{\partial x^i} \cdot \frac{\partial}{\partial x^j}.$$

The *contravariant metric tensor* g^{ij} is defined as the inverse of the covariant metric tensor

$$g^{ij} g_{jk} = \delta^i_k$$

Lemma 2.7. *The metric tensor* g_{ij} *is a covariant tensor of order two.*

Proof. In definition 2.16 we called g_{ij} a covariant tensor without justification. We will now formally proof this. To that end, we need to show that g_{ij} transforms according to eq. (2.57). By eq. (2.59) we have that $\mathbf{X}_{i'} = \mathbf{X}_i J^i_{i'}$ and obtain

$$g_{i'j'} = \mathbf{X}_{i'} \cdot \mathbf{X}_{j'} = \mathbf{X}_i J^i_{i'} \cdot \mathbf{X}_j J^j_{j'} = \left(\mathbf{X}_i \cdot \mathbf{X}_j\right) J^i_{i'} J^j_{j'} = g_{ij} J^i_{i'} J^j_{j'}.$$

$\square$

With the help of the metric tensor we can state the dot product in coordinates as $\mathbf{U} \cdot \mathbf{V} = g_{ij} u^i v^j$. Note that this holds for *any* coordinate system. The covariant metric tensor encodes complete information about the dot product in coordinates and is the central object to measure lengths, areas, volumes and angles. For instance, we compute the length of the coordinate representation of a vector in an arbitrary coordinate system as

$$|\mathbf{U}| = \sqrt{\mathbf{U} \cdot \mathbf{U}} = \sqrt{g_{ij} u^i u^j}. \tag{2.60}$$

Example 2.3. Consider the setup of example 2.2. We are going to use eq. (2.60) to compute the length of the tangent vector v from its representation in polar and Cartesian coordinates.

First, we compute the metric tensor for the polar coordinate system. The two basis vectors are orthogonal, and since the off-diagonal entries of the metric tensor are given by the dot product between the two vectors, they are zero. The diagonal entries are computed from the dot product of each of the basis vector with itself. $\frac{\partial}{\partial_r}$ is unit-length, which determines the first diagonal entry to 1. The length of $\frac{\partial}{\partial_\theta}$ is $|r|$ and since the polar coordinates of p are given by $(2, \pi)$ we have that the second diagonal entry is $|r| \cdot |r| = 4$.

Hence the metric tensor for the polar coordinate system at the point $(2, \pi)$ is

$$g_{ij}^{\text{polar}} = \begin{bmatrix} 1 & 0 \\ 0 & r^2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$$

To compute the metric tensor for the Cartesian coordinate system, we note again that the basis vectors are orthogonal. Moreover, both basis vectors are unit length, which trivially determines the metric tensor to the identity matrix

$$g_{ij}^{\text{cart}} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Now compute the length of v from its representation in polar coordinates, $v^i = (-3, 1)$

$$\|v\| = \sqrt{g_{ij}^{\text{polar}} v^i v^j} = \sqrt{1 \cdot (-3) \cdot (-3) + 4 \cdot 1 \cdot 1} = \sqrt{13}$$

The representation of v in Cartesian coordinates is $v^i = (3, 2)$ and

$$\|v\| = \sqrt{g_{ij}^{\text{cart}} v^i v^j} = \sqrt{1 \cdot 3 \cdot 3 + 1 \cdot 2 \cdot 2} = \sqrt{13}$$

The power of this concept unfolds once one considers general manifolds. Geometry is the measurement of lengths, and the metric tensor tells how to do that on an arbitrary manifold M. At every point $p \in M$ the metric tensor can be thought of as a ruler, and because the coordinate charts are diffeomorphisms the metric depends smoothly on p. Using the ruler, one can do calculus, e.g. integrate curves, measure curvature etc. On a n-dimensional manifold the metric tensor can be expressed as $n \times n$ matrix, and since the dot product between vectors is symmetric, we have that the matrix of the metric tensor is symmetric, i.e. $g_{ij} = g_{ji}$.

The metric tensor is very useful in converting covariant indices to contravariant and vice-versa. In particular, we have the relation

$$T_i = g_{ij} T^j, \tag{2.61}$$

which is called *lowering an index*. Likewise, a contraction with the contravariant metric tensor results in *raising an index*

$$T^i = g^{ij} T_j. \tag{2.62}$$

Note that since the metric tensor is symmetric, it does not make a difference on which index we contract and eq. (2.61) could be written equivalently as $T_i = g_{ji} T^j$.

We now consider an important property of the *zero-order tensor* U, namely the fact that U is an invariant, i.e. it evaluates the same in all coordinate systems: $U = U'$. By definition, an object that is independent from the coordinate system carries geometric meaning. How to obtain a zero-order tensor? By contraction, which "sums away" a covariant and a contravariant index.[11] For example, consider the tensor expression

$$U = T^i S_i,$$

which we can express in the primed coordinate system as follows

$$U' = T^i J_i^{i'} S_k J_{i'}^k = T^i S_k J_i^{i'} J_{i'}^k. \tag{2.63}$$

However, by eq. (2.56) there is an inverse relationship between the Jacobians, namely we have $J_i^{i'} J_{i'}^k = \delta_i^k$, therefore we write eq. (2.63) as

$$U' = T^i S_k \delta_i^k.$$

Now it does not matter into which tensor we absorb the Kronecker delta, in any case we get

$$U' = T^i S_i = T^k S_k = U,$$

which shows that U evaluates indeed the same in the primed coordinate system. We have already used this fact in example 2.3, where we contracted away all indices with the help of the metric tensor to obtain the length of the tangent vector v independently from the coordinate system. Thus we have a simple way to recognize geometric properties: contract away all indices and the result will evaluate the same in any coordinate system. While artifacts of the coordinate system are present in the components of a tensor, they are present in a systematic way. We have a principled method to disentangle the artifacts of the coordinate system and the geometric meaning of an expression.

2.3.5 Calculus on Manifolds

We now transfer the two elementary operations of calculus to manifolds: Integrating and taking the gradient of a function. To that end, we first introduce *vector fields* and the concept of *covectors*.

2.3.5.1 Vector Fields and Covector Fields

Definition 2.17. A *vector field* X *on a manifold* M *assigns to each point* $p \in M$ *a tangent vector* $X_p \in T_pM$.

We can think of a vector field in the usual intuitive way: It assigns an arrow to every point on the manifold. For example, let $(U, (x^i))$ be a chart on M, then the local basis

[11] Here we assert the fact that contraction of a tensor is again a tensor without formal proof.

$\frac{\partial}{\partial x^i}\big|_p$ generates a vector field as p varies over U. We call this vector field the *i-th coordinate vector field.*

Recalling theorem 2.1, a tangent vector at p is a derivation. Therefore we can use a vector field X and a smooth function $f \in C^\infty(\mathbb{R}^n)$ to construct a new function Xf by

$$(Xf)(p) = X_p f, \tag{2.64}$$

which of course means to take the directional derivative along the vector X_p of f.

Definition 2.18. A *covector* on a real vector space V is a real-valued linear functional on V. I.e. a covector ω is a linear map $\omega : V \to \mathbb{R}$. The space of all covectors is the dual space of V, denoted by V^*.

A basis $\{e_i\} \in V$ for V gives rise to a dual basis which we temporarily denote by $\{\lambda^i\} \in V^*$. The relation between the bases vectors is captured by the following fundamental equation

$$\lambda^i(e_j) = \delta^i_j. \tag{2.65}$$

For example, a vector $v \in V$ can be represented by $v = v^i e_i$, where $v^i \in \mathbb{R}$. As a result of eq. (2.65), the i-th basis covector picks out the i-th component when fed a vector v, $\lambda^i(v) = v^i$. Conversely, every covector ω can be represented in the dual basis as $\omega = \omega_i \lambda^i$. Note that components of vectors have upper indices, whereas components of covectors have lower indices. This is consistent with the conventions about contravariant and covariant tensors, see section 2.3.4.

In linear algebra, this duality has much resemblance to the relation between column and row vectors. If we represent vectors as columns, we can think of row vectors as linear functionals that act on a vector by the scalar product. In particular, it is easy to see that the row vectors $e^1 = (1\ 0\ \ldots), e^2 = (0\ 1\ 0\ \ldots)\ \ldots$ pick out the i-th component of a vector, they correspond to the (standard) dual basis. Due to this connection, the action of a covector ω on a vector v is often denoted by $\langle \omega, v\rangle = \omega(v)$. Meaning is dictated by context: Whenever one variable is a covector and the other is a vector, the scalar product of the two is understood as the action of the former on the latter.

Tangent Covectors on Manifolds On a manifold M, we have seen that we can construct at each point p the tangent space T_pM by partial differentiation of the coordinate mapping. Similar to definition 2.18, we define the *cotangent space* at p as the dual space to T_pM, denoted T_p^*M.[12] Elements ω_p of the cotangent space are called *cotangent vectors* or *covectors*. A *covector field* ω on a manifold M is a function that assigns a covector

[12]We may also use the more explicit notation $(T_pM)^*$ or $(T_p(M))^*$. Note that the subscript p is crucial, since TM and T^*M usually denote the tangent and cotangent bundle respectively. These are the collections of all tangent resp. cotangent spaces parameterized by M. In modern Riemannian geometry, the tangent space and cotangent space at a point are obtained as *sections* of the tangent and contangent bundle respectively.

ω_p to each point $p \in M$. In that sense, covector fields are dual to vector fields on M. Covector fields are also called *differential one-forms* or just *one-forms*. In eq. (2.64) we have defined the function Xf for a vector field X and smooth function f. In the dual, we can accordingly construct a covector field $f\omega$ by

$$(f\omega)_p = f(p)\omega_p. \tag{2.66}$$

One-forms belong to the more general class of differential n-forms, which play a central role in manifold theory. Whereas a one-form assigns to each point a functional which takes a tangent vector as input and returns a number, a n-form assigns a functional which takes n tangent vectors as input and returns a number.

Given a chart $(U, (x^i))$ on M, the local basis for the tangent space is given by $\left\{e_i := \frac{\partial}{\partial x^i}\big|_p\right\}$, with corresponding dual basis $\lambda^i|_p$. As in the case of a vector field, we call the covector field generated by moving p on U the *coordinate covector field*. A covector $\omega_p \in T_p^*M$ can be represented as $\omega_p = \omega_i \lambda^i|_p$, where $\omega_i = \omega_p\left(\frac{\partial}{\partial x^i}\big|_p\right)$.

2.3.5.2 The Gradient of a Function on a Manifold

Let us now turn to the gradient of a function $f \in C^\infty(M)$ on a manifold M. In standard calculus, the gradient is defined as a vector field, with the components being the partial derivatives of f. Using our framework, this would read

$$\nabla f = \sum_{i=1}^{n} \frac{\partial f}{\partial x^i}\frac{\partial}{\partial x^i}. \tag{2.67}$$

However, in this form the gradient is not independent of the coordinate system, as hinted by the fact that eq. (2.67) violates the indexing convention.

Whereas it is not possible to interpret the partial derivatives of a real-valued function as the components of a vector field in a coordinate-independent way, it turns out that they can be interpreted naturally as the components of a covector field. Given a tangent vector $v_p \in T_p\mathbb{R}^n$, we therefore define a covector field $\mathrm{d}f$ called differential of f

$$\mathrm{d}f(v_p) = \mathrm{d}f_p(v_p) = v_p f. \tag{2.68}$$

It is easy to see that $\mathrm{d}f_p$ depends linearly on v at each point $p \in M$, therefore $\mathrm{d}f_p$ is indeed a covector as defined in definition 2.18.

Let us see how $\mathrm{d}f$ looks in coordinates (x^i). With the help of the coordinate vector field $\frac{\partial}{\partial x^i}\big|_p$ and the corresponding coordinate covector field $\lambda^i|_p$ we can write $\mathrm{d}f$ as

$$\mathrm{d}f_p = \frac{\partial f}{\partial x^i}(p)\lambda^i|_p. \tag{2.69}$$

This means that the components of $\mathrm{d}f$ in any coordinate chart are given by the partial derivatives of f w.r.t. those coordinates. In particular, we can apply eq. (2.69) to the coordinate functions x^j themselves

$$\mathrm{d}x^j|_p = \frac{\partial x^j}{\partial x^i}(p)\lambda^i|_p = \delta^j_i \lambda^i|_p = \lambda^j|_p. \tag{2.70}$$

This shows that the dual basis vectors are nothing else than the differentials $\mathrm{d}x^j$*!*

Therefore we drop our preliminary notation of using λ^i for the basis covector field and modify eq. (2.65) accordingly to get the well known duality

$$\mathrm{d}x^i|_p \left(\left. \frac{\partial}{\partial x^j} \right|_p \right) = \delta^i_j, \tag{2.71}$$

which results in the following representation of a covector $\omega_p = \omega_i \mathrm{d}x^i|_p$.

If we let x, y, z be coordinates on $\mathbb{R}^3$, then $\mathrm{d}x, \mathrm{d}y, \mathrm{d}z$ are one-forms on $\mathbb{R}^3$. This gives a well-defined meaning to the widely used (and often not properly justified) notation in elementary calculus. A quote from the great Michael Spivak summarizes the state of affairs:

> Classical differential geometers (and classical analysts) did not hesitate to talk about “infinitely small” changes dx^i of the coordinates x^i, just as Leibnitz had. No one wanted to admit that this was nonsense, because true results were obtained when these infinitely small quantities were divided into each other (provided one did it in the right way).
>
> Eventually it was realized that the closest one can come to describing an infinitely small change is to describe a direction in which this change is supposed to occur, i.e., a tangent vector. Since df is supposed to be the infinitesimal change of f under an infinitesimal change of the point, df must be a function of this change, which means that df should be a function on tangent vectors. The dx^i themselves then metamorphosed into functions, and it became clear that they must be distinguished from the tangent vectors $\frac{\partial}{\partial x^i}$. Once this realization came, it was only a matter of making new definitions, which preserved the old notation, and waiting for everybody to catch up. [Spivak, 1999]

In a one-dimensional space with the coordinate x, the above reduces to the following familiar expression for the differential of a function f

$$\mathrm{d}f = \frac{\mathrm{d}f}{\mathrm{d}x}\mathrm{d}x. \tag{2.72}$$

Having an expression of the differential as a covector field, it is a simple matter of using the metric tensor to raise the index on $\mathrm{d}f$ (see eq. (2.62)), thus converting the covector field to a vector field. The components of this vector field are the partial derivatives of f, which, by definition, is nothing else than the gradient of f

$$\nabla f = g^{ij} \mathrm{d}f = g^{ij} \frac{\partial f}{\partial x^i} \frac{\partial}{\partial x^j}. \tag{2.73}$$

It is important to note that if our manifold is $\mathbb{R}^n$ with a Cartesian coordinate system, the metric tensor is given by the identity matrix, i.e. it is a flat space, and eq. (2.73) reduces to the standard definition of the gradient. In this case there is no distinction between vectors and covectors, since essentially they all live in the same space, i.e. vector fields and covector fields can be used interchangeably. On a general manifold however, vectors and covectors live in different spaces (cf. section 2.3.2.2) and have to be treated on their own terms. The metric tensor is the fundamental object to achieve this, it defines the inner product on the tangent space in a way such that everything “works correctly”.

Pullback of Covector Fields In section 2.3.2.3 we have established that a smooth map $F : M \to N$ between manifolds M, N induces a linear map $\mathrm{d}F_p : T_pM \to T_{F(p)}N$ between the tangent spaces at every point $p \in M$. We called this map the *differential* or *pushforward* of F. We can see the relation between this differential and the differential as a covector field from the previous section eq. (2.68) by considering eq. (2.49) and the special case that $F : M \to \mathbb{R}$. Under the canonical identification of $\mathbb{R}$ with $T_{F(p)}\mathbb{R}$, the two definitions are really the same. Thus we are justified in calling both of them the differential.

The differential pushes tangent vectors at p from M to N, and it is natural to consider the dual of the differential, called *codifferential*, which is a linear map between the cotangent spaces

$$\mathrm{d}F_p^* : T^*_{F(p)}N \to T^*_pM. \tag{2.74}$$

The codifferential is defined by

$$\left(\mathrm{d}F_p^*(\omega_{F(p)})\right)(v) = \omega_{F(p)}(\mathrm{d}F_p(v)), \tag{2.75}$$

where $\omega_{F(p)} \in T^*_{F(p)}N$ and $v \in T_pM$. Intuitively, the codifferential reverses the direction and pulls back a covector at $F(p)$ from N to M. If ω is a covector field on N, we call

$$(F^*\omega)_p = \mathrm{d}F_p^*(\omega_{F(p)}) \tag{2.76}$$

the *pullback* of the covector $\omega_{F(p)}$ by F, i.e. the pullback F^* is the codifferential. In particular, the pullback commutes with the differential and the product as shown in the next proposition.

Proposition 2.1. *Let $F : M \to N$ be a smooth mapping between manifolds M, N, $h \in$*

$C^\infty(N)$ a real-valued function on N and ω a covector field on N. Then

$$F^*(h\omega) = (h \circ F)F^*\omega \tag{2.77}$$
$$F^*\mathrm{d}h = \mathrm{d}(h \circ F) \tag{2.78}$$

Proof. To proof eq. (2.77), it suffices to check that for any $p \in M$

$$\begin{aligned}
(F^*(h\omega))_p &= \mathrm{d}F_p^*\left((h\omega)_{F(p)}\right) && \text{(by the pullback eq. (2.76))}\\
&= \mathrm{d}F_p^*\left(h(F(p))\omega_{F(p)}\right) && \text{(by eq. (2.66))}\\
&= h(F(p))\mathrm{d}F_p^*(\omega_{F(p)}) && \text{(by linearity of } \mathrm{d}F_p^*\text{)}\\
&= h(F(p))(F^*\omega)_p && \text{(by the pullback eq. (2.76))}\\
&= ((h \circ F)F^*\omega)_p && \text{(by eq. (2.66))}
\end{aligned}$$

To proof eq. (2.78) let $v \in T_pM$ be a tangent vector and compute

$$\begin{aligned}
(F^*\mathrm{d}h)_p(v) &= \left(\mathrm{d}F_p^*(\mathrm{d}h_{F(p)})\right)(v) && \text{(by eq. (2.76))}\\
&= \mathrm{d}h_{F(p)}\left(\mathrm{d}F_p(v)\right) && \text{(by definition of } \mathrm{d}F_p^* \text{ eq. (2.75))}\\
&= \mathrm{d}F_p(v)h && \text{(by definition of the differential eq. (2.68))}\\
&= v(h \circ F) && \text{(by definition of the differential eq. (2.50))}\\
&= \mathrm{d}(h \circ F)_p(v) && \text{(by definition of the differential eq. (2.68))}
\end{aligned}$$

□

We note that a real-valued function $h \in C^\infty(N)$ on N can be pulled back by

$$F^*h = h \circ F. \tag{2.79}$$

2.3.5.3 Integration on $\mathbb{R}^n$

Concerning integration, there is no way to directly integrate a function on a manifold in a coordinate-independent way. Consider for example any closed ball $C \subset \mathbb{R}^n$ and the function $f : C \to \mathbb{R}$ which is equal to 1, $f(x) \equiv 1$. Then the integral

$$\int_C f\mathrm{d}V = Vol(C)$$

with $\mathrm{d}V$ being the volume element is clearly not invariant under coordinate transformations. The theory of integration on manifolds is extraordinarily rich, and we can give only a very brief overview of the most important concepts here. The subsequent exposition closely follows the very insightful article “Differential Forms and Integration” by Terence Tao.[13]

[13]`http://www.math.ucla.edu/~tao/preprints/forms.pdf`

Let us start with the most basic situation to integrate a smooth real-valued function in a one-dimensional space. For instance, we are interested in the amount of work it takes to move a one-dimensional particle from a to b in the presence of a force field given by a function $f : \mathbb{R} \to \mathbb{R}$, i.e. to compute the integral $\int_a^b f(x)\mathrm{d}x$. We can approximate this infinitesimally by considering the work to move the particle from a position $x_i \in \mathbb{R}$ to a nearby position $x_{i+1} \in \mathbb{R}$.[14] This work will be (up to small errors) linearly proportional to the displacement $\Delta x_i := x_{i+1} - x_i$, with the proportionality constant given by $f(x_i)$, i.e. the work is given by $f(x_i)\Delta x_i$. We now select any discrete path $x_0 = a, x_1, x_2, \dots, x_n = b$ between a and b and approximate the integral as

$$\int_a^b f(x)\mathrm{d}x \approx \sum_{i=0}^{n-1} f(x_i)\Delta x_i.$$

If we now let the maximum step size $\sup\{|\Delta x_i|,\ 0 \leq i \leq n-1\}$ go to zero, eventually the discrete sum will converge to the value of the continuous integral on the left hand side.

Let us now consider the more advanced case to integrate over a path from $a \in \mathbb{R}^n$ to $b \in \mathbb{R}^n$ in a n-dimensional ambient space. Such path can be described as a space curve $\gamma : [0,1] \to \mathbb{R}^n$, with $\gamma(0) = a$ and $\gamma(1) = b$. This canonical description of a space curve is called a *parameterization.* If the curve is given in a different form, for example as a function from some interval $[r,s]$ to the ambient space, it can always be reparameterized into the canonical form. Now the positions on our discrete path are points $x_i \in \mathbb{R}^n$, e.g. with coordinate t on the interval $[0,1]$ we have $x_0 = a = \gamma(0), x_1 = \gamma(t_1), x_2 = \gamma(t_2), \dots, x_n = b = \gamma(1)$ where $t_i \in [0,1]$. This makes the displacements Δx_i into *vectors*, more precisely $\Delta x_i \in T_{x_i}\mathbb{R}^n$, i.e. the displacements are tangent vectors to $\mathbb{R}^n$ at the point x_i. In the one-dimensional case, we converted the displacement Δx_i via the proportionality constant $f(x_i)$ into a new number $f(x_i)\Delta x_i$. In higher dimensions, we will thus need a (linear) transformation which takes a tangent vector Δx_i and returns a number, i.e. a functional $\omega_{x_i} : \mathbb{R}^n \to \mathbb{R}$. The suggestive notation ω_{x_i} is on purpose, since it is clear that what we are looking for is exactly a covector, see definition 2.18. If we let ω be a differential one-form on $[a,b]$, that is, a covector field, we can compute the work to move from a to b along γ as

$$\int_\gamma \omega \approx \sum_{i=0}^{n} \omega_{x_i}(\Delta x_i).$$

This shows that differential forms are "the natural thing" to integrate on a manifold.

Let us see how this connects to the usual notation of the line integral in the one-dimensional case. With the standard coordinate x on $\mathbb{R}$, a one-form ω can be written as $\omega_x = f(x)\mathrm{d}x$ for some smooth function $f : [a,b] \to \mathbb{R}$. Then we define the integral of ω

[14] here i is simply a running index, it does not denote components of a (co)vector.

over $[a,b]$ as

$$\int_{[a,b]} \omega = \int_a^b f(x)\mathrm{d}x. \tag{2.80}$$

This construction is more than just a trick of notation. If $\varphi : [c,d] \to [a,b]$ is an increasing diffeomorphism, meaning that $t_1 < t_2$ implies $\varphi(t_1) < \varphi(t_2)$, then we have that

$$\int_{[c,d]} \varphi^* \omega = \int_{[a,b]} \omega,$$

which shows eq. (2.80) is invariant under diffeomorphisms (if φ is instead a decreasing diffeomorphism, there is a simple sign change involved).

In a higher-dimensional space, the above concepts generalize in the following way. Let $D \subset \mathbb{R}^n$ be a domain of integration, and ω a differential n-form, i.e. a functional that takes n vectors and returns a number. Similar to the one-dimensional case, such n-form can be written as $\omega = f\mathrm{d}x^1 \wedge \cdots \wedge \mathrm{d}x^n$, where $\wedge$ denotes the wedge product and $f : D \to \mathbb{R}$ is some function. The integral of ω over D is defined as

$$\int_D \omega = \int_D f\mathrm{d}x^1 \wedge \cdots \wedge \mathrm{d}x^n,$$

which is often written more suggestively by erasing the wedges as

$$\int_D \omega = \int_D f\mathrm{d}x^1 \cdots \mathrm{d}x^n.$$

One has to take care when using this last notation. Without the wedges it is easy forget that the integrand is a n-form. In particular, this notational convenience only works when the n-form is specified as above, as a wedge product of one-forms in increasing order. If this is not the case, there is an additional sign change involved. The reason for this is the basic fact that the wedge product is anticommutative. If ω is a k-form and ν a l-form, then $\omega \wedge \nu = (-1)^{kl} \nu \wedge \omega$. In particular, for one-forms $\mathrm{d}x^1, \mathrm{d}x^2$ this means that $\mathrm{d}x^1 \wedge \mathrm{d}x^2 = -\mathrm{d}x^2 \wedge \mathrm{d}x^1$ and as a result $\int_D f\mathrm{d}x^1 \wedge \mathrm{d}x^2 = -\int_D f\mathrm{d}x^2 \wedge \mathrm{d}x^1$.

The equivalent to invariance under diffeomorphisms from the one-dimensional case is obtained as follows. Let $E \subset \mathbb{R}^n$ be another domain of integration and $G : D \to E$ a smooth map. If ω is an n-form on E

$$\int_D G^* \omega = \int_E \omega,$$

that is, the change of integration domains is obtained by pulling back ω by G.[15]

[15]To be precise, we need the additional requirement that G is orientation-preserving. We skip the technical details of what exactly this means and just mention that if G is orientation-reversing, then the integral on the right hand side changes sign.

2.3.5.4 Integration on Manifolds

On a n-dimensional manifold M with a chart (U, φ), the integral of a n-form ω is defined as

$$\int_M \omega = \int_{\varphi(U)} \left(\varphi^{-1}\right)^* \omega. \tag{2.81}$$

Now consider a different coordinate mapping ψ with the same U and the fact that the transition map $\varphi \circ \psi^{-1} : \psi(U) \to \varphi(U)$ is a diffeomorphism. Then

$$\int_{\varphi(U)} \left(\varphi^{-1}\right)^* \omega = \int_{\psi(U)} \left(\varphi \circ \psi^{-1}\right)^* \left(\varphi^{-1}\right)^* \omega = \int_{\psi(U)} \left(\psi^{-1}\right)^* \omega,$$

which shows that integrating a n-form on a manifold is indeed independent of the choice of coordinates.

We now mention a very important property of a Riemannian manifold, that is, a manifold with the structure of a metric g. On such a manifold there exists a unique n-form $\mathrm{d}V$, sometimes also denoted $\mathrm{d}V_g$ or ω_g, which satisfies

$$\mathrm{d}V(e_1, \dots, e_n) = 1 \tag{2.82}$$

for any local orthonormal coordinate frame $\{e_i\}$. In particular, the coordinate expression of $\mathrm{d}V$ in coordinates (x^i) is given as

$$\mathrm{d}V = \sqrt{\det\left(g_{ij}\right)}\,\mathrm{d}x^1 \wedge \cdots \wedge \mathrm{d}x^n. \tag{2.83}$$

This n-form is also called the *Riemannian volume element*. Its importance stems from the fact that it allows to integrate functions, not just differential forms. The following beautiful equation brings together all the concepts developed so far.

Let $f \in C^\infty(U)$ be a function on $U \subseteq M$, then $f\mathrm{d}V$ is a n-form and we define the *integral of f over U* as

$$\int_U f\mathrm{d}V = \int_{\varphi(U)} f(x)\sqrt{\det\left(g_{ij}\right)}\,\mathrm{d}x^1 \cdots \mathrm{d}x^n. \tag{2.84}$$

We point out that in $\mathbb{R}^n$ with a Cartesian coordinate system (x^i), the metric tensor is given by the identity matrix. Hence the Riemannian volume element is 1. If we let Ω be the domain of integration *given in the coordinate representation*, i.e. $\Omega = \varphi(U)$, then we recover the familiar equation

$$\int_\Omega f(x)\mathrm{d}x^1 \cdots \mathrm{d}x^n = \int_\Omega f(x)\mathrm{d}x$$

for the integral of a function over Ω.

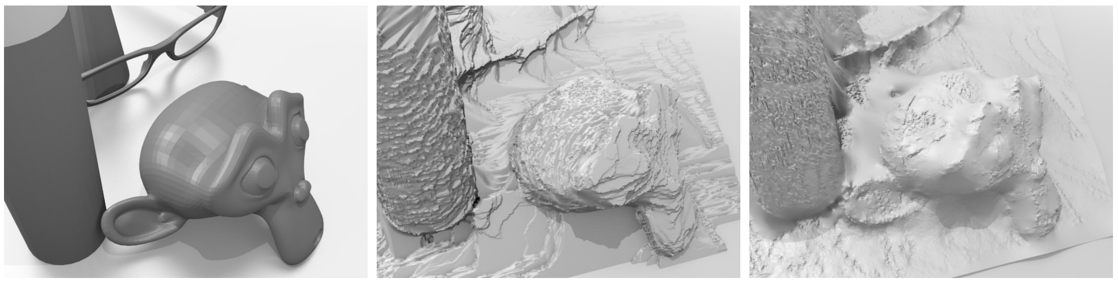
3 Metric Regularization for Surfaces

Contents

In this chapter we will address the issue of *regularization* in the context of 3D surfaces. In the general energy minimization eq. (1.4), the regularizer is the functional $R(u)$. Recalling that the energy minimization we wish to solve is ill-posed, we face the problem that there exist infinitely many possible solutions. The purpose of the regularizer is to pick one particular solution by imposing prior assumptions. It therefore plays a key role both in the quality of the solution, as well as in the efficiency of the optimization algorithm.

3.1 Minimal Area Variational Stereo Model

We consider the case where the unknown u corresponds to a 3D surface. This situation appears naturally in the 3D reconstruction problem, where the goal is to reconstruct 3D structure from the projections onto 2D images. The most principled approaches to 3D reconstruction aim to infer a collection of (multiply-connected, piecewise smooth) surfaces directly, represented intrinsically without regards to the images [Balzer and Soatto, 2014, Delaunoy and Pollefeys, 2014, Hernández and Schmitt, 2004, Pons et al., 2006, Tyleček and Šará, 2010, Zaharescu et al., 2011], as evident by the large body of literature on shape space and shape optimization. In these methods, both the geometry and the topology is then inferred to fit the available images. This is desirable as one can enforce priors on the surfaces based on physically meaningful regularizers. The disadvantage is that inferring topology is difficult and requires computation of visibility at each iteration of the algorithm, with obvious repercussions on computational efficiency.



(a) Ground truth **(b)** TV regularization **(c)** Our method

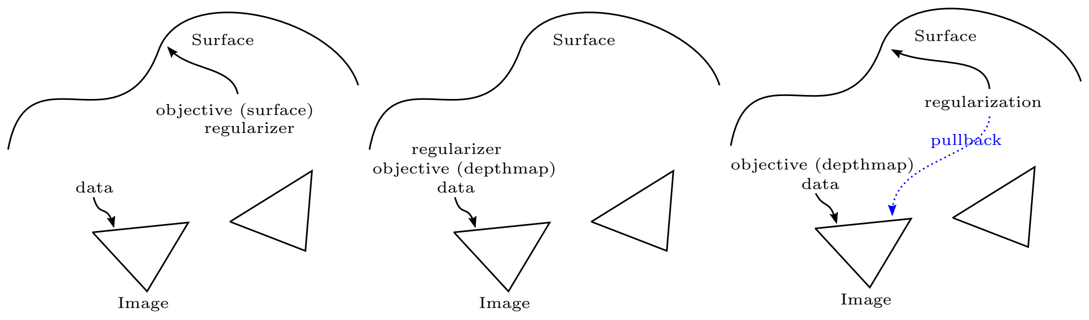
Figure 3.1: TV regularization tends to favor piecewise-constant functions, which is detrimental in the case of depth maps that represent a 3D surface. Our regularizer, while being defined on the image, just as TV, respects the inner geometry of the 3D surface.

On the other hand, one could use depthmaps that associate a positive number (distance) to each pixel [Campbell et al., 2008, Fan and Ferrie, 2010, Gallup, 2014, Goesele et al., 2006, Hu and Mordohai, 2012, Sormann et al., 2007, Tanskanen et al., 2013, Tola et al., 2011]. This approach is strongly connected to differential geometry. In particular, there is a mapping that connects the value of a pixel (its depth), which is defined on a 2D domain, to the position of a point in 3D space. This is exactly a *chart* on the surface, i.e. a depthmap is a local parameterization of a 3D surface, see section 2.3.3. Note that such depthmaps will have to be combined in an additional fusion step to yield the global surface. The advantage of using depthmaps is that they conveniently confine the data (images), the optimization variable (surfaces) and hence the objective function to the same domain, the image plane of a reference view. This makes computation efficient, and the method of choice for real-time applications. Unfortunately, the image plane is not the natural place to enforce regularization of the surface. In the vicinity of depth discontinuities, caused by occlusions, neighboring pixels do not necessarily correspond to points which are close in 3D space. Thus typical image-based regularizers, such as Total Variation (TV), favor piecewise fronto-parallel depthmaps, resulting in staircasing artifacts, see fig. 3.1(b).

3.1.1 Overview and Related Work

In the following, we seek to combine the advantages of shape space methods with those of range maps. The advantage of the former is the availability of surface-based, physically plausible regularizers, whereas the use of range maps allows us to avoid the inference of scene topology. To the best of our knowledge, this has not been done in the literature on variational stereo and is made possible by a number of technical contributions summarized in section 3.1.1.

In addition to variational reconstruction algorithms described thus far, the research on 3D reconstruction spawned a variety of methods that seek to bypass the complexities of computing topology or visibility by localizing the surface



(a) Direct surface regularization **(b)** Image-based depthmap regularization **(c)** Image-based *surface* regularization

Figure 3.2: 3D reconstruction from images with direct surface regularization (a). Topology has to be taken into account explicitly, which makes optimization hard. 3D reconstruction by parameterized depthmaps (b) uses efficient image-based regularization. The regularizer is disconnected from properties of the surface. Our minimal area regularizer (c) is image-based, but regularizes a physically meaningful property of the surface.

representation to subsets of the image plane. These image patches are small enough that correspondence with a topologically-connected surface patch can be maintained [Bailer et al., 2012, Besse et al., 2012, Bodis-Szomoru et al., 2014, Bleyer et al., 2011, Bradley et al., 2008, Furukawa and Ponce, 2010, Klaus et al., 2006, Raposo et al., 2014, Wang and Zheng, 2008]. Often, the optimization is restricted to a collection of small planar facets rotating and moving along the viewing rays of the reference camera, a process referred to as *plane sweeping*. In the variational setting that we adopt here, the object of inference, including the depthmap, is a *function*. This distinguishes our approach from patch-based methods which we will therefore consider no further. The majority of variational methods resort to *implicit* handling of depth discontinuities by TV regularization [Graber et al., 2011, Newcombe et al., 2011, Liu et al., 2009, Stühmer et al., 2010, Wendel et al., 2012].

While TV effectively handles depth discontinuities in images, it does not impose geometrically meaningful constraints on the depthmap: In section 3.1.3.2, we show that TV is a proxy of the minimal-area functional *provided the depth map is orthographic*, a rather unrealistic assumption. Straightforward coupling of TV with a perspective reprojection error ceases to be physically plausible and yields undesirable staircasing artifacts, see fig. 3.1, fig. 3.2(b). Therefore, we wish to design image-based regularizers that impose a geometrically meaningful prior on the surface, see fig. 3.2(c). In section 3.1.3.3, we derive the correct area form for the perspective case and embed it in a novel regularization term for variational stereo. While this makes the regularizer plausible, it makes the resulting optimization challenging. Thus, our second goal is to devise an efficient optimization method tailored to this regularizer. On this topic, our core contribution is to to re-parametrize the regularizer into a form that is linear in the optimization variable and thus amenable to highly efficient primal-dual solvers. This is made possible by the gauge freedom in the parametrization: we exploit the fact that there are infinitely many equivalent parameterizations to our advantage. The approach is summarized in fig. 3.2. The implementation details of our method are provided in section 3.1.4.2 and section 3.1.4.3. A series of experiments on synthetic and real data confirm our theoretical findings and demonstrate a gain in reconstruction quality, section 3.1.5.

In two companion papers [Li and Zucker, 2006a, Li and Zucker, 2006b], Li and Zucker recognize the need for richer geometric representations in stereo vision. Their work has initiated a series of enhancements of patch-based methods [Besse et al., 2012, Bleyer et al., 2011, Taniai et al., 2014, Woodford et al., 2009, Zhang et al., 2014] that all include some crude approximation of surface curvature in the proposed energy functional. Recently, Heise et al. [Heise et al., 2013] proposed to augment the PatchMatch algorithm with a term reminiscent of a Huber norm applied to normal changes across different patches. All of the aforementioned approaches depart from a discrete, label-based formulation of the problem, whose solution is accomplished by combinatorial optimization. Combinatorial optimization is however contrary to the calculus of variations, which we have chosen as our paragon here. Weighing the advantages and disadvantages of both paradigms against each other is beyond the scope of this paper, but we believe that the latter offers more flexibility in accurately modelling the inner geometry of regular surfaces. In the variational setting, Total Generalized Variation (TGV) has helped to diminish staircasing by enriching the piecewise constant basis that spans the space of functions of minimal TV with polynomials of higher order [Heber and Pock, 2014, Ranftl et al., 2012]. Still, TGV is a generic regularizer not specifically designed for surface parametrization, unlike the regularizer introduced here. Re-parametrizations of range maps to the benefit of optimization have appeared previously, e.g., in the realm of shape from shading [Prados and Faugeras, 2005] or self-localization and mapping [Civera et al., 2008].

3.1.2 Epipolar Geometry

If we have two images of a scene and know corresponding points, it is possible to reconstruct 3D structure. The underlying principle is called *epipolar geometry*, it describes the relations between a 3D point and its projection into images. We see that this is an instance where the search for correspondences appears in an important low-level vision problem. While it is certainly possible, and often necessary, to have multiple images of a scene in order to do reconstruction, the canonical case of two images is of major importance. It is the topic of a research area known as stereo matching.

We give here a short overview of epipolar geometry and introduce the concepts that are necessary to develop the variational stereo model in the next section. A more in-depth treatment can be found e.g. in [Hartley and Zisserman, 2004].

The n-dimensional projective space $\mathbb{P}^n$ is defined as the quotient space of the equivalence relation

$$\begin{aligned}(x^1, \dots, x^{n+1}) &\sim (\hat{x}^1, \dots, \hat{x}^{n+1}) \\ \text{iff } \exists \alpha \neq 0 : (x^1, \dots, x^{n+1}) &= \alpha(\hat{x}^1, \dots, \hat{x}^{n+1}),\end{aligned}$$

i.e. two points are equivalent if they are the same up to a non-zero scale factor. Such points are called *homogeneous*, and the last $(n+1)$-th component is designated as the homogeneous weight. The relation between projective coordinates and the usual Euclidean coordinates is given by $x_{\text{Euclid}} = (x^1, \dots, x^n) \leftrightarrow x_{\text{proj}} = (x^1, \dots, x^n, 1)$. Given a point in projective space with homogeneous weight $\neq 0$, one obtains Euclidean coordinates by dividing all components through the homogeneous weight and discarding the homogeneous weight. We will denote this operation by

$$\begin{aligned}\pi : \mathbb{P}^n &\to \mathbb{R}^n \\ (x^1 \dots, x^{n+1}) &\mapsto (x^1/x^{n+1}, \dots, x^n/x^{n+1}),\end{aligned} \tag{3.1}$$

with the inverse mapping

$$\begin{aligned}\pi^{-1} : \mathbb{R}^n &\to \mathbb{P}^n \\ (x^1, \dots, x^n) &\mapsto (x^1, \dots, x^n, 1).\end{aligned} \tag{3.2}$$

Note that this means loss of information. The homogeneous weight of a point in projective space cannot be recovered from its Euclidean representation. The $(n+1)$-th component is lost through the application of the mapping π.

Let us define

$$K = \begin{bmatrix} f_1 & 0 & c_1 \\ 0 & f_2 & c_2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.3}$$

the *intrinsic calibration matrix* of the camera. The parameters $f_1, f_2 \in \mathbb{R}$ are the focal length and $c_1, c_2 \in \mathbb{R}$ represents the principal point. This camera model is known as the *pinhole model*, it describes the relationship between two-dimensional points on the image plane and three-dimensional points on the $Z = 1$ plane of the camera. In particular, let $X = (X, Y, Z)^T$ be a 3D point, then its projection onto the image plane is given as

$$x = (x, y)^T = \varphi(K, X), \tag{3.4}$$

where the mapping $\varphi : \mathbb{P}^2 \to \mathbb{R}^2 : X \mapsto \pi(KX)$ combines the pinhole model and transformation to non-homogeneous coordinates. By slight abuse of notation we overload the symbols X, x to mean both the vector and its first component. Usually this will not be a problem, in case there is ambiguity we will explicitly resolve it. In the above equation the multiplication $KX \in \mathbb{P}^2$ yields a point in the two-dimensional projective space, represented as a vector in $\mathbb{R}^3$.

Given a pixel coordinate $x = (x, y)^T$, we find the corresponding 3D point at the $Z = 1$ plane by the inverse mapping $\varphi^{-1} : \mathbb{R}^2 \to \mathbb{P}^2$ given by

$$X = (X, Y, 1) = \varphi^{-1}(K, x) = K^{-1}\pi^{-1}(x) \,. \tag{3.5}$$

Note that this backprojection tells us nothing about the actual 3D position of a given image pixel. We only recovered the 3D coordinates at the $Z = 1$ plane. We also emphasize that the backprojected point is given in the local camera coordinate frame with the optical center of the camera as the coordinate origin.

Camera position and orientation in the global world coordinate frame are described by the *extrinsic parameters*, which can be represented as a matrix $C \in \mathrm{SE}(3)$, the special Euclidean group of $\mathbb{R}^3$. Elements of SE(3) can be further decomposed into a rotation matrix $R \in \mathrm{SO}(3)$, the special orthogonal group, and a translation vector $t \in \mathbb{R}^3$. The special orthogonal group of $\mathbb{R}^3$ is given by the orthogonal 3×3 matrices, i.e. matrices which fulfill $RR^T = I$, where I denotes the identity matrix. A useful consequence is that the inverse rotation is easy to compute via $R^{-1} = R^T$. The extrinsic parameters are then given as the matrix $C = [R \,|\, t] \in \mathbb{R}^{3\times4}$. Note that we are working in the projective space $\mathbb{P}^3$, where a homogeneous 3D point X is given by a vector in $\mathbb{R}^4$, therefore the matrix multiplication CX makes sense. We can always augment the extrinsic matrix with an additional row of the zero vector and homogeneous weight 1, which yields the 4×4 matrix

$$C = \left[\begin{array}{ccc|c} & R & & t \\ \hline 0 & 0 & 0 & 1 \end{array}\right].$$

It is important to know in which direction the matrix C transforms a 3D point X. We will adopt the convention that

$$X_{\text{cam}} = CX_{\text{world}},$$

multiplication with C transforms a point from the global world coordinate frame into the local camera coordinate frame. In the local camera frame the origin $(0, 0, 0, 1)^T$ is defined as the optical center of the camera, and we can obtain the position of the camera in the global frame by $C^{-1}(0, 0, 0, 1)^T = [R^{-1} \,|\, -R^{-1}t](0, 0, 0, 1)^T = -R^T t$.

Recalling the backprojection eq. (3.5), we can construct a ray[1] between center of the camera and each backprojected point, given by

$$X(\alpha) = \alpha X = (\alpha X, \alpha Y, \alpha),\ \ \alpha \geq 0. \tag{3.6}$$

We know that the actual 3D point must lie somewhere on this ray, i.e. there is some $\hat{\alpha}$ for

[1]here we are working in the local camera frame, i.e. camera center is the origin

which $\hat{\alpha}X = (\hat{\alpha}X, \hat{\alpha}Y, \hat{\alpha})^T$ is the position of the 3D point corresponding to pixel x. We call the value $\hat{\alpha}$ the *depth* of pixel x. If we represent X in projective space, we obtain the interesting relationship that the homogeneous weight corresponds to inverse depth. The ray can be alternatively given by

$$X(\alpha) = \left(X, Y, 1, \tfrac{1}{\alpha}\right), \tag{3.7}$$

and using eq. (3.1) it is easy to check that this is equivalent to eq. (3.6).

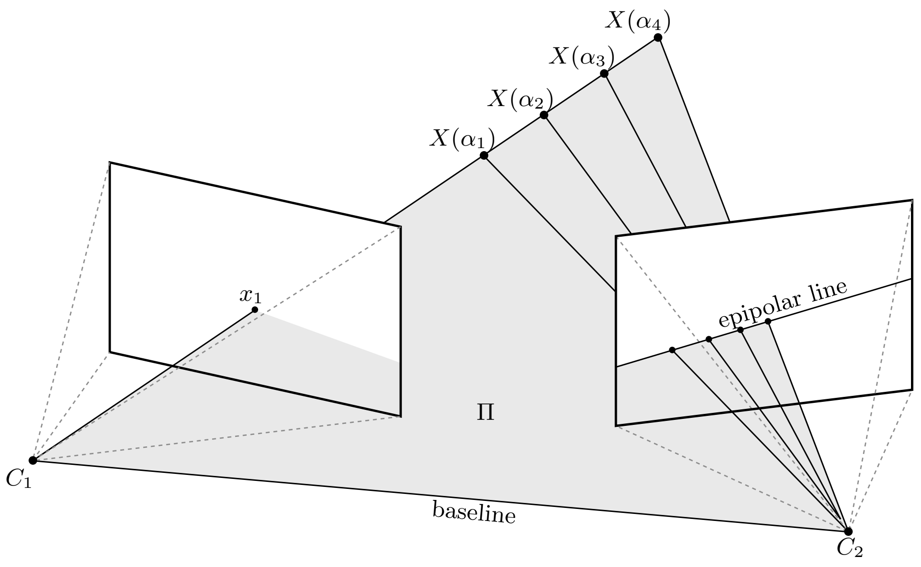


Figure 3.3: Epipolar Geometry. An image point x_1 in the first image gives rise to an epipolar line in the second image. The epipolar line is obtained by intersecting the epipolar plane Π with the image plane. Π is spanned by the basline and the viewing ray $X(\alpha)$. Note that the epipolar geometry is independent of the scene geometry, i.e. the actual depth of the 3D point corresponding to x_1 does not matter.

Let us now assume that we have two cameras C_1, C_2 observing the scene, with corresponding images I_1, I_2. We wish to establish a constraint on corresponding image points. To that end, assume a pixel x_1 in the first image backprojects to a ray $X(\alpha)$. Furthermore we can construct the line between the centers of the two cameras, called *baseline*. The backprojected ray and the baseline span a plane in space, called the *epipolar plane* Π. The intersection of this plane with the image plane of the second camera yields the *epipolar line* l_2 corresponding to pixel x_1. Thus we get the constraint that the pixel corresponding to x_1 must lie somewhere on l_2 in the second image, see fig. 3.3. By the classic duality between points and lines in projective space,[2] the epipolar line can be written as

$$l_2 = F x_1, \tag{3.8}$$

[2]One associates to a point with projective coordinates $(a, b, c)^T \in \mathbb{P}^2$ the line $ax + by + cz = 0$.

where the 3×3 matrix F is the *fundamental matrix*. From the duality between points and lines, a point x is on the line l iff $l^T x = 0$. Since we know that the point x_2 corresponding to x_1 must lie on the epipolar line, we have $l_2^T x_2 = 0$. Inserting eq. (3.8), we get the epipolar constraint between corresponding points

$$l_2^T x_2 = (F x_1)^T x_2 = x_1^T F^T x_2 = x_2^T F x_1 = 0. \tag{3.9}$$

This constraint is independent of scene geometry: Assuming known intrinsics, the epipolar plane is completely determined by the baseline and the viewing ray $X(\alpha)$ corresponding to x_1, both of which can be computed from the extrinsic camera parameters. In particular, the actual depth of the 3D point corresponding to x_1 is not needed. This means that if the intrinsics are known, the fundamental matrix can be computed from the extrinsic parameters $C_1, C_2 \in \mathrm{SE}(3)$ alone. The search for correspondence then reduces to a one-dimensional problem, as the corresponding point is constrained by eq. (3.9) to lie on the epipolar line.

Vice-versa, if we know corresponding points x_1, x_2, we can compute their rays $C_1^{-1} X_1(\alpha), C_2^{-1} X_2(\beta)$, where multiplication with the inverse extrinsic parameters transforms the ray from the local camera coordinate frame to the global world coordinate frame. The intersection of the rays then uniquely determines the position of the 3D point $X = C_1^{-1} X_1(\hat{\alpha}) = C_2^{-1} X_2(\hat{\beta})$ corresponding to the image points x_1, x_2, with $\hat{\alpha}$ being the depth of X as seen from the first image, and $\hat{\beta}$ the depth as seen from the second image.

3.1.3 Variational Stereo Model

In the following we will develop our variational stereo model by considering the canonical binocular stereo problem. An extension to more than two views is conceptually straightforward.

3.1.3.1 Data Term

We model the relative position and orientation of the sensors by an element $G_{\mathrm{rel}} \in \mathrm{SE}(3)$. Two radiance images $\mathcal{I}_r, \mathcal{I} : \Omega \to \mathbb{R}_+$, the former being the *reference* image and $\Omega \subset \mathbb{R}^2$ denoting the image domain, give rise to the *re-projection residual*

$$r = \mathcal{I} \circ w(x, z(x)) - \mathcal{I}_r(x). \tag{3.10}$$

The domain warping $w = \varphi \circ G_{\mathrm{rel}} \circ \varphi^{-1}$ is obtained by projecting pixels $x \in \Omega$ back onto the surface $S \subset \mathbb{R}^3$ and then into the projection center of the camera displaced by G_{rel}. Finally, the data term accumulates some robust Huber function $|\cdot|_\epsilon$ of the re-projection error over S:

$$E(S) = \int_\Omega |r|_\epsilon \, \mathrm{d}\boldsymbol{x}. \tag{3.11}$$

Note that this integral is computed over Ω which rules out trivial solutions such as $S = \emptyset$ that may occur in shape-space-based methods discussed in section 3.1.1. Also note that the back-projection φ^{-1} needed to compute the warping w depends on the – initially unknown – surface S, and thus also on its parameterization which we will turn to now.

3.1.3.2 Orthographic Minimal-Area Regularizer

In the simplest case where φ is orthographic, we can model S by the graph of a scalar function $z : \Omega \to \mathbb{R}$. Each point $X \in S$ can then be written as

$$X = \varphi(x) = \begin{pmatrix} x \\ y \\ z(x, y) \end{pmatrix}. \tag{3.12}$$

Note that conceptually, the surface has not much in common with the depth map, yet z appears in the parametrization and thus will influence the inner geometry of S. The metric tensor (see definition 2.16) at a point $X = \varphi(x)$ reads

$$\mathbf{I} = \begin{bmatrix} \langle X_x, X_x \rangle & \langle X_x, X_y \rangle \\ \langle X_x, X_y \rangle & \langle X_y, X_y \rangle \end{bmatrix}, \tag{3.13}$$

where the tangent vectors $X_x = \frac{\partial \varphi}{\partial x}$ and $X_y = \frac{\partial \varphi}{\partial y}$ are obtained by partial differentiation w.r.t. x and y. In the context of surfaces embedded in 3D space, the metric tensor is also known as *first fundamental form*, a term which dates back to Gauss. Typically, $\mathbf{I}$ is used to measure infinitesimal lengths and angles on the surface. In particular, $\sqrt{\det(\mathbf{I})}$ determines the distortion of the two infinitesimal area elements $\mathrm{d}x$ and $\mathrm{d}S$. A scalar function $f : S \to \mathbb{R}$ defined on the surface can be pulled back by φ to the parametric domain Ω. The pullback also relates the domains of integration S and Ω with each other:

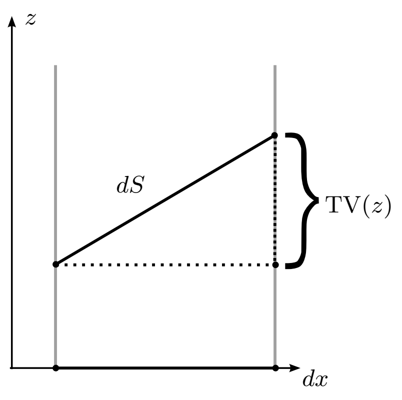
$$\int_S f(X)\mathrm{d}S = \int_\Omega f \circ \varphi(x) \sqrt{\det(\mathbf{I})}\,\mathrm{d}x. \tag{3.14}$$

Setting $f = 1$ and substituting eq. (3.12) into eq. (3.13) and then eq. (3.14) yields the total area

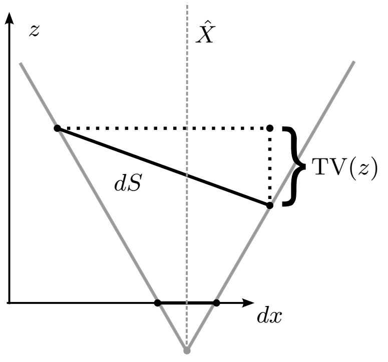
$$A(z) = \int_\Omega \sqrt{\det(\mathbf{I})}\,\mathrm{d}x = \int_\Omega \sqrt{z_x^2 + z_y^2 + 1}\,\mathrm{d}x \tag{3.15}$$

of the graph of z.

Recall that $\mathrm{TV}(z) := \int_\Omega |Dz|\mathrm{d}x$, where D denotes the distributional derivative which makes the definition well-defined even for non-differentiable functions. If z is differentiable, we have $\mathrm{TV}(z) = \int_\Omega |\nabla z|\mathrm{d}x = \int_\Omega \sqrt{z_x^2 + z_y^2}\,\mathrm{d}x$. Note that eq. (3.15) looks much like the TV of z if it were not for the additional value 1 under the square root. It is this difference that allows measuring the area of a surface element. Still, both the area form eq. (3.15)



(a) Orthographic



(b) Perspective

Figure 3.4: TV and surface area under orthographic projection (a) differ in that the TV measures only the jumps along z, whereas a surface element $\mathrm{d}S$ also takes into account the component of $\mathrm{d}\boldsymbol{x}$ parallel to the image plane. To reduce the area of a non-minimal surface element, the only option in both cases is to rotate until fronto-parallelity is achieved, i.e. $\mathrm{TV}(z) = 0$. The area form induced by perspective projection (b) on the other hand has an additional degree of freedom: the area of a non-minimal surface element can either be decreased by rotating it until perpendicular to the pixel viewing ray, or by moving it closer to the center of projection.

and $\mathrm{TV}(z)$ have the tendency to create piecewise fronto-parallel surfaces, as can be seen in fig. 3.4(a).

This property may be desirable in image processing applications, where z corresponds to some image intensity distribution over Ω, but certainly not when z is a depthmap that parameterizes a geometric surface. Meanwhile, the assumption of an orthographic camera model in reconstruction is quite unrealistic for practical purposes, and so is the use of $\mathrm{TV}(z)$ as a regularizer, although that appears to be common practice in previous works [Graber et al., 2011, Newcombe et al., 2011, Liu et al., 2009, Stühmer et al., 2010, Wendel et al., 2012]. So in the following section, let us clarify how the area form of a perspective depthmap parametrization looks like, and highlight its interplay with the TV.

3.1.3.3 Perspective Minimal-Area Regularizer

In the perspective parameterization, for which – to reduce the notational burden – we maintain the symbol φ, the depth z influences all three spatial coordinates of a surface point. More precisely, we have $X = z\hat{X}$, where

$$\hat{X} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ 1 \end{pmatrix} = \varphi^{-1}(x) = \begin{pmatrix} \frac{x-c_1}{f_1} \\ \frac{y-c_2}{f_2} \\ 1 \end{pmatrix} \tag{3.16}$$

is the direction of the viewing ray associated with a pixel. From the tangent vectors

$$X_x = \begin{pmatrix} \hat{x} z_x + \frac{z}{f_1} \\ \hat{y} z_x \\ z_x \end{pmatrix}, \quad X_y = \begin{pmatrix} \hat{x} z_y \\ \hat{y} z_y + \frac{z}{f_2} \\ z_y \end{pmatrix}, \tag{3.17}$$

both, the fundamental form and the square root of its determinant, immediately follow:

$$\sqrt{\det(\mathbf{I})} = \frac{z}{f_1 f_2} \sqrt{\|\nabla_f z\|^2 + (\langle \nabla_f z, \hat{x} \rangle + z)^2}. \tag{3.18}$$

Here, we have introduced an abbreviation ∇_f for the nabla operator whose components are weighted by the focal lengths $f = (f_1, f_2)$, i.e., $\nabla_f z = (f_1 z_x, f_2 z_y)$.

Equation (3.18) presents the central ingredient in our regularization term, so a few remarks are in order: First of all, the factor z in front of the square root makes the surface area form $\sqrt{\det(\mathbf{I})}\,\mathrm{d}S$ distance-dependent. Consequently, a mean curvature flow is furnished with an additional degree of freedom. As shown in fig. 3.4(b), one can reduce surface area (locally) by moving points towards the center of projection. Vice-versa, minimal surfaces, unless they constitute the global optimum $z = 1$ at which $\mathrm{d}S = \mathrm{d}x$, are not necessarily piecewise constant in depth. As we will later verify empirically, this helps reduce staircasing artifacts generated by methods with “naive” TV regularization. On the downside, eq. (3.18) is neither equal to the TV nor to the norm of some linear operator applied to z. Hence, a model combining the data term eq. (3.11) and area form under perspective projection eq. (3.18) does not yield a functional of the class eq. (2.20). The culprit is the area form eq. (3.18), which is non-convex. We will address this issue in the next section by transforming our variational problem such that eq. (3.18) becomes tractable for the primal-dual solver algorithm 2.5.

3.1.4 Optimization

3.1.4.1 Algorithm

Let us first summarize the continuous variational stereo model we wish to solve:

$$\min_z \int_\Omega \frac{z}{f_1 f_2} \sqrt{|\nabla_f z|^2 + (\langle \nabla_f z, \hat{x} \rangle + z)^2}\, \mathrm{d}x + \lambda \int_\Omega |\mathcal{I} \circ w(x, z(x)) - \mathcal{I}_r(x)|_\epsilon \, \mathrm{d}x. \tag{3.19}$$

As customary, $\lambda \in \mathbb{R}_+$ is a scalar parameter controlling the trade-off between data fidelity and smoothness. A similar approach was used recently by the authors of [Stühmer et al., 2010] to compute dense depthmaps in real-time, albeit employing the raw TV for regularization. Flow-based stereo carries out a continuous search for correspondences along the epipolar lines. It can thus be seen as the variational counterpart of the planesweep algorithm, however, with the advantages that it requires no resource-hungry spatial data structure. Also, the extension to multiple views is easy to achieve by summing up the reprojection error over a number of image pairs.

Looking at the first term of eq. (3.19), we see that in general it is non-convex because of the bilinear form involving z and its derivative. If we use the fact that $\sqrt{\det(\mathbf{I})}$ equals the length of the surface normal $\|n\| = \|X_x \times X_y\|$, the situation improves slightly. From eq. (3.17), it becomes clear, though, that

$$n = \begin{pmatrix} -\frac{zz_x}{f_2} \\ -\frac{zz_y}{f_1} \\ \frac{1}{f_1 f_2}(z^2 + \hat{x} f_1 z z_x + \hat{y} f_2 z z_y) \end{pmatrix} \tag{3.20}$$

and therefore $\|n\|$ in general is still non-convex in z. Remarkably, this can be fixed by re-parameterizing S as stated in the central

Proposition 3.1. *Substituting z in the perspective depth map parameterization by $z = \phi(\zeta)$ with $\phi(\zeta) = \sqrt{2\zeta}$, the surface normal becomes a linear function of ζ. In particular, it holds*

$$n(\zeta) = \begin{pmatrix} -\frac{\zeta_x}{f_2} \\ -\frac{\zeta_y}{f_1} \\ \frac{\hat{x}\zeta_x}{f_2} + \frac{\hat{y}\zeta_y}{f_1} + \frac{2\zeta}{f_1 f_2} \end{pmatrix}. \tag{3.21}$$

Proof. We start by applying the chain rule to the non-convex term zz_x (similarly to zz_y), which yields for a re-parameterization $z = \phi(\zeta)$

$$zz_x = \phi \frac{\mathrm{d}\phi}{\mathrm{d}\zeta} \frac{\partial \zeta}{\partial x}. \tag{3.22}$$

If we now require that

$$\phi \frac{\mathrm{d}\phi}{\mathrm{d}\zeta} = 1, \tag{3.23}$$

we are left with the term $\frac{\partial\zeta}{\partial x}$, which can be expressed as a linear operator on ζ. Equation (3.23) constitutes a first-order ordinary differential equation, which can be solved for ϕ by *separation of the variables*:

$$\begin{aligned} \phi \mathrm{d}\phi &= \mathrm{d}\zeta \\ \int \phi \mathrm{d}\phi &= \int \mathrm{d}\zeta \\ \frac{\phi^2}{2} &= \zeta. \end{aligned} \tag{3.24}$$

From eq. (3.24), we get $\phi = \sqrt{2\zeta}$. Inserting this into eq. (3.20) and using $\frac{\mathrm{d}\phi}{\mathrm{d}\zeta} = \frac{1}{\phi}$, it follows immediately that

$$
\begin{aligned}
zz_x &= \phi \frac{\mathrm{d}\phi}{\mathrm{d}\zeta} \frac{\partial d\zeta}{\partial x} = \sqrt{2\zeta} \frac{\mathrm{d}\sqrt{2\zeta}}{\mathrm{d}\zeta} \zeta_x = \zeta_x, \\
zz_y &= \phi \frac{\mathrm{d}\phi}{\mathrm{d}\zeta} \frac{\partial \zeta}{\partial y} = \zeta_y, \\
z^2 &= \phi^2 = 2\zeta,
\end{aligned}
\tag{3.25}
$$

and hence the claim. □

Note that the transformation $\zeta = \frac{z^2}{2}$ can be interpreted as a change of coordinates, see section 2.3.2.3. We have effectively re-parameterized the variational problem on the manifold given by the parabola $\frac{z^2}{2}$, where it turns out to be convex. This is an example of the maxim stated at the beginning of our treatment of differential geometry in section 2.3: The coordinate system is dictated by the problem at hand, and sometimes an appropriate choice of coordinates can make a problem easier to analyze and solve.

Let us remark that the transformation ϕ is bijective and differentiable over $(0, \infty)$, which is sufficient since we may assume all surface points to be located in front of the camera.

We are now left with the non-convexity of the data term. Since the optimization variable z appears as an argument to the warping w in eq. (3.19), it is clear the only way to get around the non-convexity is to linearize the data term. This calls for an iterative optimization strategy, in which at each step, say $k \in \mathbb{N}$, a local convex approximation of the data term is minimized. With the residual

$$
r(z) = \mathcal{I} \circ w(x, z(x)) - \mathcal{I}_r(x), \tag{3.26}
$$

the idea is to compute a Taylor expansion

$$
r(z) \approx r(z_k) + \frac{\mathrm{d}r}{\mathrm{d}z}\Big|_{z_k} (z - z_k) \tag{3.27}
$$

around the current iterate z_k, yielding a local approximation of eq. (3.11):

$$
\hat{E}(z) := \int_\Omega \left| r(z_k) + \frac{\mathrm{d}r}{\mathrm{d}z}\Big|_{z_k} (z - z_k) \right|_\epsilon \mathrm{d}x. \tag{3.28}
$$

The overall strategy is reminiscent of the classical Gauss-Newton method for nonlinear least-squares problems. The last piece we need in order to finalize treatment of the data term is the derivative of the residual w.r.t. the function z, which is given by

$$
\frac{\mathrm{d}r}{\mathrm{d}z} = \nabla \mathcal{I}|_{\varphi \circ G_{\mathrm{rel}}(z\hat{X})} D\varphi|_{G_{\mathrm{rel}}(z\hat{X})} DG_{\mathrm{rel}} \hat{X}, \tag{3.29}
$$

where the operator D denotes differentiation. To conclude this section, let us state the full variational problem in the variable ζ following re-parametrization:

$$\min_{\zeta} \int_{\Omega} |n(\zeta)|\, \mathrm{d}x + \lambda \int_{\Omega} |\mathcal{I} \circ w(x, \zeta(x)) - \mathcal{I}_r(x)|_{\epsilon}\, \mathrm{d}x. \tag{3.30}$$

Note that since the original warp w depends on z, it also has to be reformulated in terms of ζ.

3.1.4.2 Discretization

We discretize the image domain Ω on the regular Cartesian grid of size $M \times N$ and represent images as matrices in $\mathbb{R}^{M \times N}$. To discretize the gradient, we utilize linear operators $D_x, D_y : \mathbb{R}^{M \times N} \to \mathbb{R}^{M \times N}$ corresponding to finite difference approximations of the partial derivatives in x and y direction. Their action on an image $u \in \mathbb{R}^{M \times N}$ is given by [Chambolle, 2004]

$$(D_x u)_{i,j} = \begin{cases} u_{i,j+1} - u_{i,j} & \text{if } j < N, \\ 0 & \text{else,} \end{cases}$$
$$(D_y u)_{i,j} = \begin{cases} u_{i+1,j} - u_{i,j} & \text{if } i < M, \\ 0 & \text{else.} \end{cases}$$

Now we can define a linear operator $L : \mathbb{R}^{M \times N} \to \mathbb{R}^{M \times N \times 3}$, which computes for every element of $\zeta \in \mathbb{R}^{M \times N}$ its normal vector according to eq. (3.21) as follows

$$\begin{aligned} (L\zeta)_{ij1} &= \frac{(D_x\zeta)_{ij}}{f_2} \\ (L\zeta)_{ij2} &= \frac{(D_y\zeta)_{ij}}{f_1} \\ (L\zeta)_{ij3} &= \frac{\hat{x}_{ij}(D_x\zeta)_{ij}}{f_2} + \frac{\hat{y}_{ij}(D_y\zeta)_{ij}}{f_1} + \frac{2}{f_1 f_2}, \end{aligned} \tag{3.31}$$

where $\hat{x}, \hat{y} \in \mathbb{R}^{M \times N}$ are images of the x and y components of the viewing ray eq. (3.16). The discrete regularization term is then given by $\|L\zeta\|_{1,1,2}$, where the $1, 1, 2$-norm is obtained by taking the ℓ_1-norm along the first two dimensions and the ℓ_2-norm along the third. That is, for $A \in \mathbb{R}^{M \times N \times K}$ we have $\|A\|_{1,1,2} = \sum_{i=1}^{M} \sum_{j=1}^{N} \sqrt{\sum_{k=1}^{K} (A_{ijk})^2}$. We note that the operator L can be represented as a matrix $\bar{L} \in \mathbb{R}^{3MN \times MN}$, acting by matrix-vector multiplication on the vectorized image $\bar{\zeta} \in \mathbb{R}^{MN \times MN}$. Then $(\bar{L}\bar{\zeta}) \in \mathbb{R}^{3MN}$, which can be rearranged in the obvious way, i.e. inverse to the vectorization operation, to obtain $(L\zeta) \in \mathbb{R}^{M \times N \times 3}$ as required by eq. (3.31).

The discrete data term is given by $\|r\|_{\epsilon}$, where $r \in \mathbb{R}^{M \times N}$ is the residual eq. (3.26). The discrete Huber norm is defined as the element-wise sum $\|r\|_{\epsilon} = \sum_i \sum_j |r_{ij}|_{\epsilon}$.

3.1.4.3 Implementation

Recalling that our strategy to solve the originally non-convex problem eq. (3.19) is to linearize the data term around some estimate ζ_k and subsequently solve a series of locally-convex approximations, each of the sub-problems looks like

$$\min_{\zeta} \|L\zeta\|_{1,1,2} + \lambda \|r + a \odot (\zeta - \zeta_k)\|_{\epsilon}, \tag{3.32}$$

where $a := \frac{\mathrm{d}r}{\mathrm{d}\zeta}\big|_{\zeta_k} \in \mathbb{R}^{M\times N}$ is the element-wise derivative of $r \in \mathbb{R}^{M\times N}$ w.r.t. $\zeta \in \mathbb{R}^{M\times N}$ and $\odot$ denotes element-wise multiplication. We point out that while eq. (3.32) and eq. (3.19) are equivalent, they differ in the important aspect that the regularizer in eq. (3.19) is non-convex in the optimization variable, whereas the regularizer of eq. (3.32) is the norm of a linear operator applied to ζ, hence it is convex. For this reason, we can apply the highly efficient first order primal-dual algorithm due to Chambolle and Pock [Chambolle and Pock, 2011] to minimize eq. (3.32). The the primal-dual formulation reads

$$\min_{\zeta} \max_{\|q\|_{\infty,\infty,2}\leq 1} \langle L\zeta, q\rangle + \lambda \|r + a \odot (\zeta - \zeta_k)\|_{\epsilon}, \tag{3.33}$$

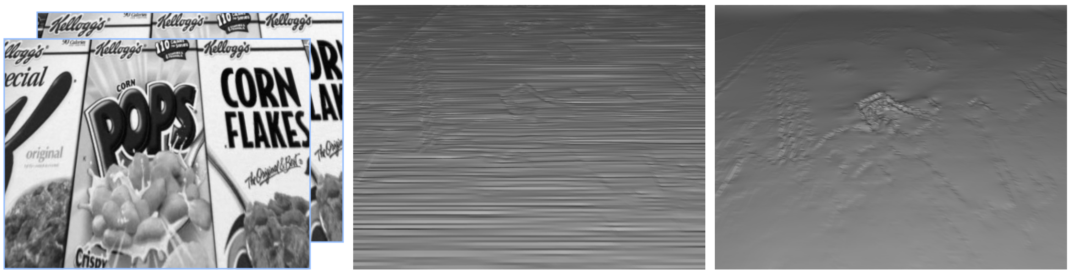
where $q \in \mathbb{R}^{M\times N\times 3}$ is the dual variable and the scalar product for arrays $A, B \in \mathbb{R}^{M\times N\times K}$ is defined in the obvious way as $\langle A, B\rangle = \sum_{i=1}^{M}\sum_{j=1}^{N}\sum_{k=1}^{K} A_{ijk}B_{ijk}$. The inner iteration – denoted by l to distinguish it from the outer iterations k – delivers a minimizer of eq. (3.33) with convergence rate $\mathcal{O}(l)$

$$\begin{aligned}
q^{l+1} &= \mathrm{proj}_{\|q\|_{\infty\,\infty,2}\leq 1}(q^l + \Sigma \odot (L\tilde{\zeta}^l)), && (3.34)\\
\zeta^{l+1} &= \mathrm{prox}(\zeta^l - T \odot (L^* q^{l+1})), && (3.35)\\
\tilde{\zeta}^{l+1} &= 2\zeta^{l+1} - \zeta^l,
\end{aligned}$$

where L^* denotes the adjoint operator of L. We can see from eq. (3.21) that L has irregular structure: For a fixed index (i, j) it computes a vector with 3 components, where the range of the last component is much different from the range of the first two. This makes estimation of the operator norm difficult, and likely it will be large, meaning that the algorithm converges slowly. Therefore we apply a preconditioning, denoted by T, Σ, according to [Pock and Chambolle, 2011]. To that end, we utilize the matrix representation $\bar{L} \in \mathbb{R}^{3MN\times MN}$ of L, see section 3.1.4.2. We compute preconditioning vectors τ, σ by

$$\tau_n = \frac{1}{\sum_{m=1}^{3MN} |\bar{L}_{mn}|}, \qquad \sigma_m = \frac{1}{\sum_{n=1}^{MN} |\bar{L}_{mn}|}.$$

Rearranging elements, we get $T \in \mathbb{R}^{M\times N} \Leftrightarrow \tau \in \mathbb{R}^{MN}$ and $\Sigma \in \mathbb{R}^{M\times N\times 3} \Leftrightarrow \sigma \in \mathbb{R}^{3MN}$, and T, Σ can be multiplied element-wise with $(L^*q) \in \mathbb{R}^{M\times N}$ and $(L\tilde{\zeta}) \in \mathbb{R}^{M\times N\times 3}$ respectively in eq. (3.35) and eq. (3.34).



(a) Input image pair **(b)** 3D surface (TV) **(c)** 3D surface (ours)

Figure 3.5: Results for the synthetic experiment consisting of a tilted plane to show the behavior of the regularizer in case of a slanted surface. The TV regularizer (b) produces the well-known staircaising artifacts while the surface area regularizer (c) shows no bias towards fronto-parallelity.

$\text{proj}_{\|q\|_{\infty,\infty,2}\leq 1}(\cdot)$ is a simple pointwise projection onto the unit ball. The proximal operator for the data term can be solved explicitly by the following formula:

$$\hat{\zeta} = \text{prox}(\xi) \Leftrightarrow \hat{\zeta}_{ij} = \begin{cases} \xi_{ij} - T_{ij}\lambda a_{ij} & \text{if } a_{ij}\xi_{ij} + b_{ij} > T_{ij}\lambda a_{ij}^2 + \epsilon \\ \xi_{ij} + T_{ij}\lambda a_{ij} & \text{if } a_{ij}\xi_{ij} + b_{ij} < -T_{ij}\lambda a_{ij}^2 - \epsilon \\ \frac{\xi_{ij} - T_{ij}\lambda a_{ij} b_{ij}/\epsilon}{1 + T_{ij}\lambda a_{ij}^2/\epsilon} & \text{else} \end{cases}, \tag{3.36}$$

where $b := r - a \odot \zeta_k$.

As it is common in many optical flow algorithms, we embed the whole procedure into a coarse-to-fine warping framework to account for large discontinuities in depth. We created a highly parallel implementation using the CUDA toolkit, which makes the method attractive for (near) real-time applications.

3.1.5 Experimental Studies

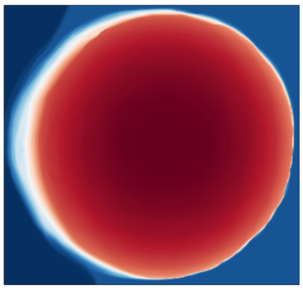
All results were computed on a desktop PC equipped with a 3.2GHz i7 QuadCore CPU and a Geforce 780Ti GPU. We used a pyramid scale factor of 0.75 throughout, and computed 30 warps per pyramid level and 60 iterations per warp.

For a real-world configuration with a scale factor of 0.5, 20 warps and 30 iterations (note that this is rarely needed for convergence and can be trimmed further), the runtime is 0.14 s for a resolution of 640×480 and 1.9 s for 3072×2048 respectively.

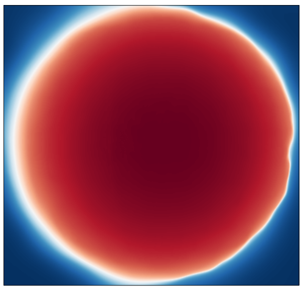
3.1.5.1 Synthetic Data

To empirically verify the theoretical properties of our regularizer (see section 3.1.3.3), we conducted a number of experiments with synthetic (i.e., perfect) input data and known ground truth. Despite the fact that our model is conceptually capable of using multiple views, we restricted ourselves to classical binocular stereo for all experiments.

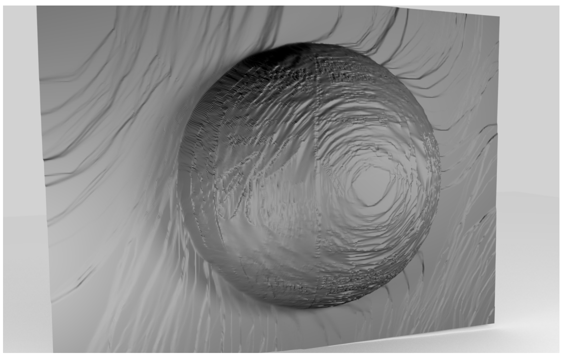
The first example consists of a plane rotated 30° around the x-axis. As depicted in



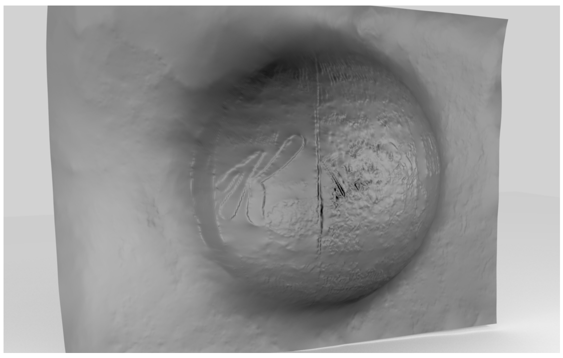
(a) Depth map TV



(b) Depth map Ours



(c) 3D surface (TV), $\lambda = 0.15$



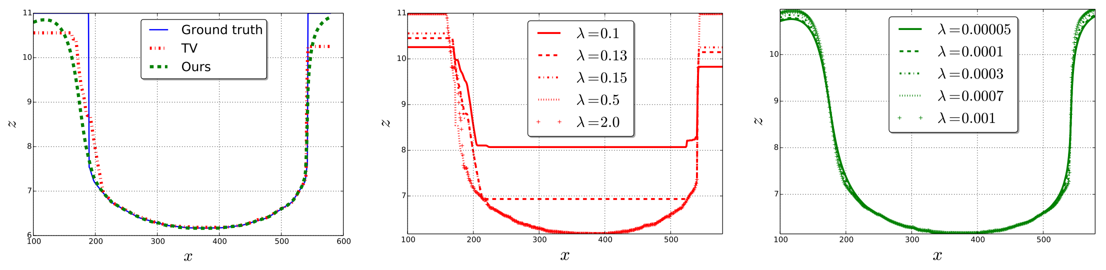
(d) 3D surface (ours), $\lambda = 3 \cdot 10^{-4}$

Figure 3.6: Results for a synthetic curved surface. Whereas the depth maps (a)-(b) of the sphere surface look similar, a 3D visualization (c)-(d) allows a more thorough examination of reconstruction quality: the plateau structure is visible in case of TV regularization (especially at the top), area regularization on the other hand is more faithful to the true curved surface.

fig. 3.5, TV regularization clearly shows staircaising artifacts, whereas our surface area regularizer produces a smooth slanted plane. We emphasize the fact that the regularization strength for TV has been hand-tuned to be as smooth as possible before breaking down (e.g., approximating the slanted surface by a series of very large fronto-parallel steps).

The next experiment (fig. 3.6) involves a hemisphere set against a fronto-parallel background. We use the hemisphere to assess the ability of the regularizer to reconstruct curved surfaces. Figure 3.6(c) and fig. 3.6(d) show a 3D visualization of the result for TV and surface area regularization respectively.Despite of the depth maps fig. 3.6(a) and fig. 3.6(b) of the sphere looking similarly smooth, one can see qualitative differences in the 3D surface: Where TV tends to approximate the half-sphere by fronto-parallel plateaus, area regularization produces a more pleasing result. This underlines the importance of using 3D visualizations when assessing the quality of stereo algorithms.

Figure 3.7 illustrates the behaviour of the regularizer under varying values of the regularization parameter λ. TV (fig. 3.7(b)) suffers from sudden breakdown whereas area regularization (fig. 3.7(c)) remains stable over orders of magnitude. Stable parameters are of great interest to the practitioner because most of these are still hand-tuned. Note that for the visualizations of the sphere (fig. 3.6(c) and fig. 3.6(d)) we again chose favorable regularization strength for TV (i.e., close to breakdown) and medium regularization for surface area.



(a) Reconstruction vs. groundtruth **(b)** TV at different levels of regularization **(c)** Ours at different levels of regularization

Figure 3.7: Horizontal cross-section through the depth maps for the sphere experiment. Note that the depth discontinuity at the left side is occluded in the input images; the right side is co-visible. We find that in occluded regions, where the task of the regularizer is hallucination, TV produces the familiar steps. Area regularization on the other hand smoothly bridges the occluded area.

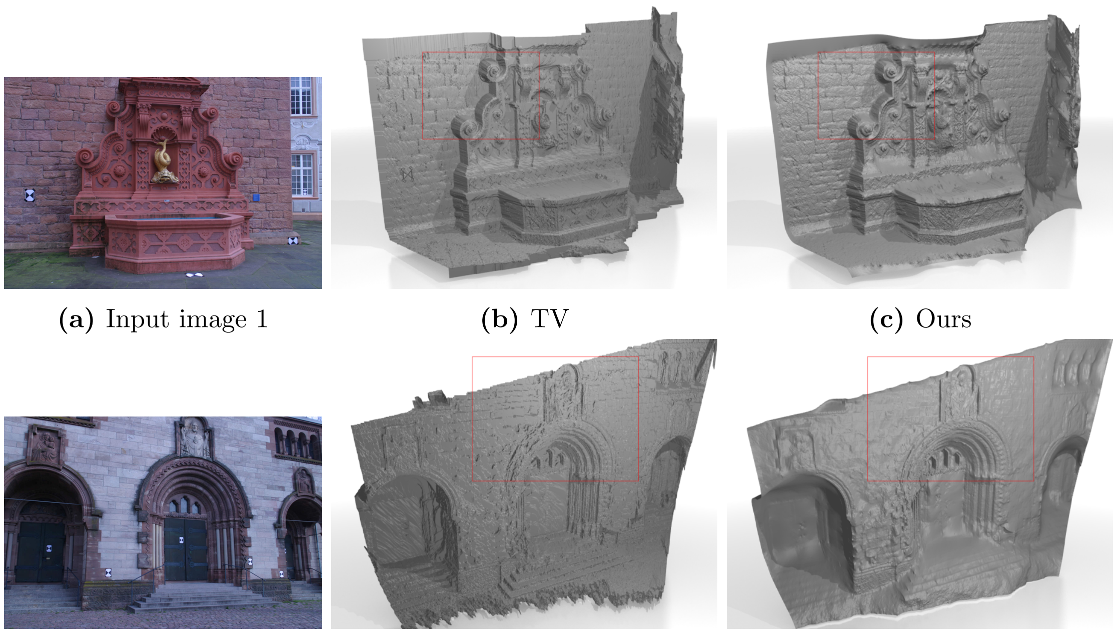
3.1.5.2 Real Data

We tested our regularizer on real-world examples taken from the Strecha dataset [Strecha et al., 2008]. It consists of a number of high-resolution (3072×2048) images for dense multiview stereo algorithms. All results were obtained using only two images. Figure 3.8 depicts 3D renderings of results from the *Fountain-P11* and *Herz-Jesu-P25* scene. We also provide the input images for reference. Despite our best efforts and a very strong weight on the data term (as can be seen in fig. 3.8(e) by the artifacts at the bottom and the top), we did not succeed in getting a smooth reconstruction of the church facade by means of TV regularization. The reason for this is that the facade is slightly slanted w.r.t. the reference camera image plane. The gaps between the individual weakly-textured bricks provide a gradient for the TV regularizer to hold on to. It therefore tends to approximate every individual brick by its own fronto-parallel facet, as can be seen in the closeup fig. 3.9(c). The surface area regularizer fig. 3.9(d) is able to recover small depth discontinuities, see for instance the little arches above the front door, while maintaining a smooth facade.

Table 3.1 shows a quantitative comparison of the Root Mean Square (RMS) error against ground-truth depthmaps over different datasets.

3.1.6 Conclusion

We have introduced a new regularizer for variational stereo, which is defined on the image but regularizes a geometrically meaningful quantity on the surface. Exploiting gauge freedom, a re-parameterization makes the regularizer compatible with highly efficient primal-



(a) Input image 1 (b) TV (c) Ours

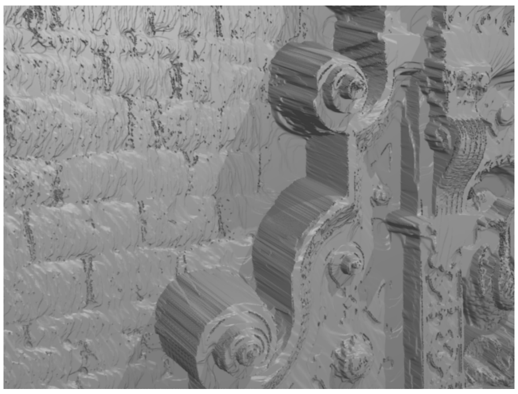
(d) Input image 1 (e) TV (f) Ours

Figure 3.8: Results for the *Fountain-P11* and *Herz-Jesu-P25* scene. Closeups can be found in fig. 3.9.

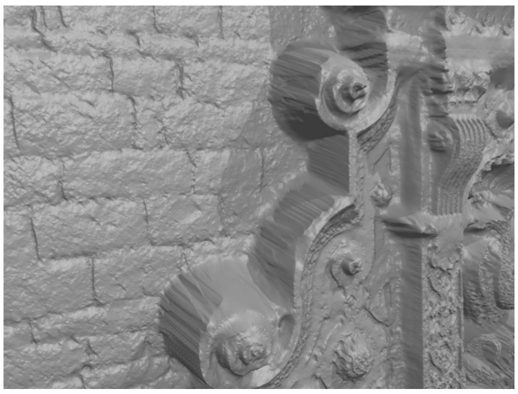
	RMS error		**Reduction**
	TV	Ours	$\frac{\text{TV}-\text{Ours}}{\text{TV}}$
TiltedPlane10	5.5527e-4	1.0819e-4	80.5%
TiltedSine10	0.0167188	0.0114407	31.6%
Fountain-P11	0.05342	0.02644	50.5%
Herz-Jesu-P25	0.264935	0.222491	16.0%

Table 3.1: *RMS* error against ground-truth depth-maps for different datasets. The last column is the error reduction, i.e., the percentaged gain in reconstruction quality achieved by our method over *TV* regularization.

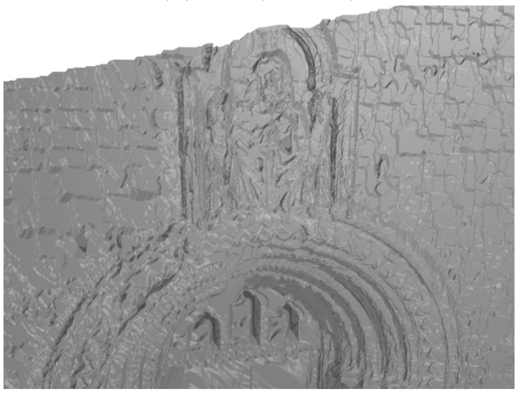
dual solvers for large scale problems. We evaluated important properties of the regularizer such as the ability to reconstruct smooth surfaces using both synthetic and real world data. In particular, a comparison to the widely used *TV* regularizer showed that the minimal surface regularizer does not suffer from the staircaising effect. Because the computational cost remains basically the same when going from *TV* to surface area regularization, our method is suited for real-time applications.



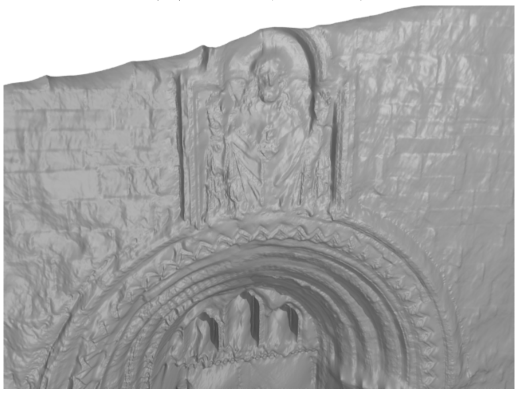
(a) TV (closeup)



(b) Ours (closeup)



(c) TV closeup

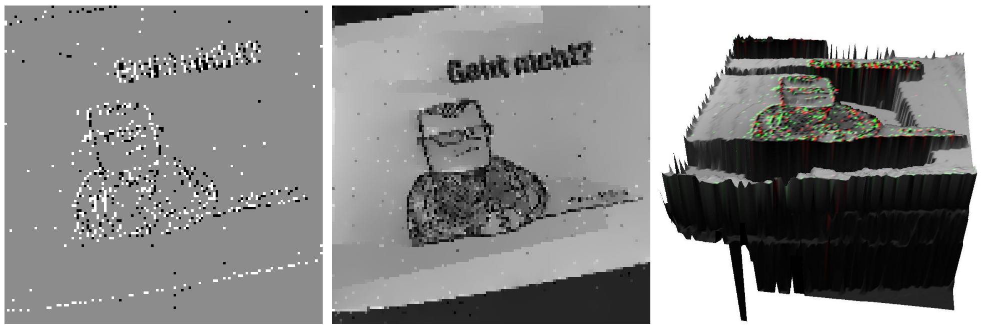


(d) Ours closeup

Figure 3.9: Closeup renderings of the results on Strecha dataset.

3.2 Image Reconstruction for Event Cameras using Manifold Regularization

In contrast to standard Complementary Metal-Oxide-Semiconductor (CMOS) digital cameras that operate on frame basis, neuromorphic cameras such as the Dynamic Vision Sensor (DVS)[Lichtsteiner et al., 2008] work asynchronously on a pixel level. Each pixel measures the incoming light intensity and fires an *event* when the absolute change in intensity is above a certain threshold (which is why those cameras are also often referred to as *event cameras*). The time resolution is in the order of μs. Due to the sparse nature of the events, the amount of data that has to be transferred from the camera to the computer is very low, making it an energy efficient alternative to standard *CMOS* cameras for the tracking of very quick movement [Delbruck and Lichtsteiner, 2007, Wiesmann et al., 2012]. While it is appealing that the megabytes per second of data produced by a digital camera can be compressed to an asynchronous stream of events, these events cannot be used directly in computer vision algorithms that operate on a frame basis. In recent years, the first algorithms have been proposed that transform the problem of camera pose estimation to this



(a) Raw Events **(b)** Reconstructed Image **(c)** Event Manifold

Figure 3.10: Sample results from our method. The image (a) shows the raw events and (b) is the result of our reconstruction. The time since the last event has happened for each pixel is depicted as a surface in (c) with the positive and negative events shown in green and red respectively.

new domain of time-continuous events e.g. [Benosman et al., 2014, Gallego et al., 2015, Kim et al., 2014, Mueggler et al., 2014, Mueggler et al., 2015, Weikersdorfer et al., 2013], unleashing the full potential of the high temporal resolution and low latency of event cameras. The main drawback of the proposed methods are specific assumptions on the properties of the scene or the type of camera movement.

Contribution In this work we aim to bridge the gap between the time-continuous domain of events and frame-based computer vision algorithms. We propose a simple method for intensity reconstruction for neuromorphic cameras (see fig. 3.10 for a sample output of our method). In contrast to very recent work on the same topic by [Bardow et al., 2016], we formulate our algorithm on an event-basis, avoiding the need to simultaneously estimate the optical flow. We cast the intensity reconstruction problem as an energy minimization, where we model the camera noise in a data term based on the *generalized Kullback-Leibler divergence*. The optimization problem is defined on a manifold induced by the timestamps of new events (see fig. 3.10(c)). We show how to optimize this energy using variational methods and achieve real-time performance by implementing the energy minimization on a Graphics Processing Unit (GPU). We release software to provide live intensity image reconstruction to all users of *DVS* cameras[3]. We believe this will be a vital step towards a wider adoption of this kind of cameras.

3.2.1 Related Work

Neuromorphic or event-based cameras receive increasing interest from the computer vision community. The low latency compared to traditional cameras make them particularly interesting for tracking rapid camera movement. Also more classical low-level computer vision problems are transferred to this new domain like optical flow estimation, or image

[3]`https://github.com/VLOGroup/dvs-reconstruction`

reconstruction as proposed in this work. In this literature overview we focus on very recent work that aims to solve computer vision tasks using this new camera paradigm. We begin our survey with a problem that benefits the most from the temporal resolution of event cameras: camera pose tracking. Typical Simultaneous Localization and Mapping (SLAM) methods need to perform image feature matching to build a map of the environment and localize the camera within [Hartmann et al., 2013]. Having no image to extract features from means, that the vast majority of visual *SLAM* algorithms cannot be readily applied to event-based data. [Milford et al., 2015] show that it is possible to extract features from images that have been created by accumulating events over time slices of 1000 ms to perform large-scale mapping and localization with loop-closure. While this is the first system to utilize event cameras for this challenging task, it trades temporal resolution for the creation of images like fig. 3.10(a) to reliably track camera movement.

A different line of research tries to formulate camera pose updates on an event basis. [Cook et al., 2011] propose a biologically inspired network that simultaneously estimates camera rotation, image gradients and intensity information. An indoor application of a robot navigating in 2D using an event camera that observes the ceiling has been proposed by [Weikersdorfer et al., 2013]. They simultaneously estimate a 2D map of events and track the 2D position and orientation of the robot. Similarly, [Kim et al., 2014] propose a method to simultaneously estimate the camera rotation around a fixed point and a high-quality intensity image only from the event stream. A particle filter is used to integrate the events and allow a reconstruction of the image gradients, which can then be used to reconstruct an intensity image by Poisson editing. All methods are limited to 3 Degrees of Freedom (DOF) of camera movement. A full camera tracking has been shown in [Mueggler et al., 2014, Mueggler et al., 2015] for rapid movement of an Unmanned Aerial Vehicle (UAV) with respect to a known 2D target and in [Gallego et al., 2015] for a known 3D map of the environment.

[Benosman et al., 2014] tackle the problem of estimating optical flow from an event stream. This work inspired our use of an event manifold to formulate the intensity image reconstruction problem. They recover a motion field by clustering events that are spatially and temporally close. The motion field is found by locally fitting planes into the event manifold. In experiments they show that flow estimation works especially well for low-textured scenes with sharp edges, but still has problems for more natural looking scenes. Very recently, the first methods for estimating intensity information from event cameras without the need to recover the camera movement have been proposed. [Barua et al., 2016] use a dictionary learning approach to map the sparse, accumulated event information to infer image gradients. Those are then used in a Poisson reconstruction to recover the log-intensities. [Bardow et al., 2016] proposed a method to simultaneously recover an intensity image and dense optical flow from the event stream of a neuromorphic camera. The method does not require to estimate the camera movement and scene characteristics to reconstruct intensity images. In a variational energy minimization framework, they concurrently recover optical flow and image intensities within a time window. They show that optical flow is necessary to recover sharp image edges especially for fast movements in the image. In contrast, in this work we show that intensities can also be recovered without explicitly estimating the optical flow. This leads to a substantial reduction of complexity: In our current implementation, we are able to reconstruct > 500 frames per second. While the method is defined on a per-event-basis, we can process blocks of events without loss in image quality. We are therefore able to provide a true live-preview to users of a neuromorphic camera.

3.2.2 Image Reconstruction from Sparse Events

We have given a time sequence of events $(e^n)_{n=1}^N$ from a neuromorphic camera, where $e^n = \{x^n, y^n, \theta^n, t^n\}$ is a single event consisting of the pixel coordinates $(x^n, y^n) \in \Omega \subset \mathbb{R}^2$, the polarity $\theta^n \in \{-1, 1\}$ and a monotonically increasing timestamp t^n.

A positive θ^n indicates that at the corresponding pixel the intensity has increased by a certain threshold $\Delta^+ > 0$ in the log-intensity space. Vice versa, a negative θ^n indicates a drop in intensity by a second threshold $\Delta^- > 0$. Our aim is now to reconstruct an intensity image $u^n : \Omega \to \mathbb{R}_+$ by integrating the intensity changes indicated by the events over time.

Taking the $\exp(\cdot)$, the update in intensity space caused by one event e^n can be written as

$$f^n(x^n, y^n) = u^{n-1}(x^n, y^n) \cdot \begin{cases} c_1 & \text{if } \theta^n > 0 \\ c_2 & \text{if } \theta^n < 0 \end{cases}, \tag{3.37}$$

where $c_1 = \exp(\Delta^+)$, $c_2 = \exp(-\Delta^-)$. Starting from a known u^0 and assuming no noise, this integration procedure will reconstruct a perfect image (up to the radiometric discretization caused by $\Delta^\pm$). However, since the events stem from real camera hardware, there is noise in the events. Also the initial intensity image u^0 is unknown and cannot be reconstructed from events alone. Therefore the reconstruction of u^n from f^n cannot be solved without imposing some regularity in the solution. We therefore formulate the intensity image reconstruction problem as the solution of the optimization problem

$$u^n = \operatorname*{arg\,min}_{u \in C^1(\Omega, \mathbb{R}_+)} \left[E(u) = D(u, f^n) + R(u) \right] , \tag{3.38}$$

where $D(u, f^n)$ is a *data term* that models the camera noise and $R(u)$ is a *regularisation term* that enforces some smoothness in the solution. In the following section we will show how we can utilize the timestamps of the events to define a manifold which guides a variational model and detail our specific choices for data term and regularization.

3.2.2.1 Variational Model on the Event Manifold

Moving edges in the image cause events once a change in logarithmic intensity is bigger than a threshold. The collection of all events $(e^n)_{n=1}^N$ can be recorded in a spatiotemporal volume $V \subset \Omega \times T$. V is very sparsely populated, which makes it infeasible to directly store it. To alleviate this problem, [Bardow et al., 2016] operate on events in a fixed time window that is sliding along the time axis of V. They simultaneously optimize for optical flow and intensities, which are tightly coupled in this volumetric representation.

Regularization Term As in [Benosman et al., 2014], we observe that events lie on a lower-dimensional manifold within V, defined by the most recent timestamp for each pixel $(x, y) \in \Omega$. A visualisation of this manifold for a real-world scene can be seen in fig. 3.10(c). [Benosman et al., 2014] fittingly call this manifold the *surface of active events*. We propose to incorporate the surface of active events into our method by formulating the optimisation *directly on the manifold*. Our intuition is, that parts of the scene that have no or little texture will not produce as many events as highly textured areas. Regularizing an image reconstructed from the events should take into account the different "time history" of pixels. In particular, we would like to have strong regularization across pixels that stem from events at approximately the same time, whereas regularization between pixels whose events have very different timestamps should be reduced. This corresponds to a grouping of pixels in the time domain, based on the timestamps of the recorded events. Solving computer vision problems on a surface is also known as *intrinsic image processing* [Lai and Chan, 2011], as it involves the intrinsic (i.e. coordinate-free) geometry of the surface, a topic studied by the field of differential geometry. Looking at the body of literature on intrinsic image processing on surfaces, we can divide previous work into two approaches based on the representation of the surface. Implicit approaches [Krueger et al., 2008, Cheng et al., 2000] use an implicit surface (e.g. through the zero level set of a function), whereas explicit approaches [Lui et al., 2008, Stam, 2003] construct a triangular mesh representation. Our method uses the same underlying theory of differential geometry, however we note that because the surface of active events is defined by the timestamps which are monotonically increasing, the class of surfaces is effectively restricted to $2\frac{1}{2}$D. This means that there exists a simple parameterization of the surface and we can perform all computations in a local euclidean coordinate frame, i.e. the image domain Ω, see section 2.3.3. In contrast to [Lai and Chan, 2011], where the authors deal with arbitrary surfaces, we avoid the need to explicitly construct a representation of the surface. This has the advantage that we can straightforwardly make use of GPU-accelerated algorithms to solve the large-scale optimization problem.

We start by defining the surface $S \subset \mathbb{R}^3$ as the graph of a scalar function $t(x, y)$ through the mapping $\varphi : \Omega \to S$

$$X = \varphi(x, y) = \begin{bmatrix} x, & y, & t(x, y) \end{bmatrix}^T, \tag{3.39}$$

where $X \in S$ denotes a 3D-point on the surface. $t(x, y)$ is simply an image that records for each pixel (x, y) the time since the last event. By corollary 2.1 the partial derivatives of the parameterization φ define a basis for the tangent space $T_X S$ at each point X of the surface S, and the dot product in this tangent space gives the *metric* of the manifold. In particular, the *metric tensor* is defined as the symmetric 2×2 matrix

$$g_{ij} = \begin{bmatrix} \langle \varphi_x, \varphi_x \rangle & \langle \varphi_x, \varphi_y \rangle \\ \langle \varphi_x, \varphi_y \rangle & \langle \varphi_y, \varphi_y \rangle \end{bmatrix}, \tag{3.40}$$

where subscripts denote partial derivatives and $\langle \cdot, \cdot \rangle$ denotes the scalar product. Starting from the definition of the parameterization eq. (3.39), straightforward calculation gives $\varphi_x = \begin{bmatrix} 1 & 0 & t_x \end{bmatrix}^T$, $\varphi_y = \begin{bmatrix} 0 & 1 & t_y \end{bmatrix}^T$ and

$$g_{ij} = \begin{bmatrix} 1 + t_x^2 & t_x t_y \\ t_x t_y & 1 + t_y^2 \end{bmatrix} \tag{3.41}$$

$$g^{ij} = \frac{1}{G} \begin{bmatrix} 1 + t_y^2 & -t_x t_y \\ -t_x t_y & 1 + t_x^2 \end{bmatrix}, \tag{3.42}$$

where g^{ij} denotes the inverse of the metric tensor and $G = \det(g_{ij})$.

Given a function $\tilde{f} \in C^1(S, \mathbb{R})$ on the manifold, the gradient of $\tilde{f}$ is characterized by $\nabla_g \tilde{f} = g^{ij} \mathrm{d}\tilde{f}$, see eq. (2.73). We will use the notation $\nabla_g \tilde{f}$ to emphasize the fact that we take the gradient of a function defined on the surface (i.e. under the metric of the manifold). $\nabla_g \tilde{f}$ can be expressed in local coordinates as

$$\nabla_g \tilde{f} = \left(g^{11} \tilde{f}_x + g^{12} \tilde{f}_y \right) \varphi_x + \left(g^{21} \tilde{f}_x + g^{22} \tilde{f}_y \right) \varphi_y, \tag{3.43}$$

Inserting the inverse metric tensor eq. (3.42) into the equation for the gradient eq. (3.43) gives an expression for the gradient of a function $\tilde{f}$ on the manifold in local coordinates

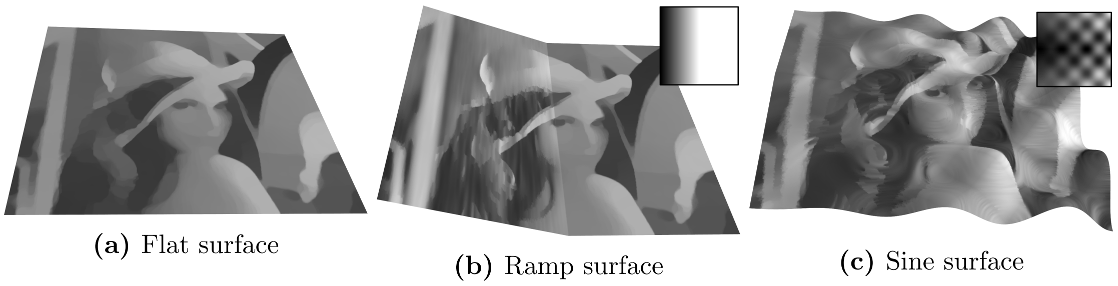
$$\nabla_g \tilde{f} = \frac{1}{G} \left\{ \left(\left(1 + t_y^2\right) \tilde{f}_x - t_x t_y \tilde{f}_y \right) \begin{bmatrix} 1 \\ 0 \\ t_x \end{bmatrix} + \left(\left(1 + t_x^2\right) \tilde{f}_y - t_x t_y \tilde{f}_x \right) \begin{bmatrix} 0 \\ 1 \\ t_y \end{bmatrix} \right\}. \tag{3.44}$$

Equipped with these definitions, we are ready to define our regularization term. It will be a variant of the TV norm insofar that we take the norm of the gradient of $\tilde{f}$ on the manifold

$$TV_g(\tilde{f}) = \int_S |\nabla_g \tilde{f}| \, \mathrm{d}s. \tag{3.45}$$

It is easy to see that if we have $t(x, y) = const$, then g_{ij} is the 2×2 identity matrix and $TV_g(\tilde{f})$ reduces to the standard TV. Also note that in the definition of the TV_g we integrate over the surface. Since our goal is to formulate everything in local coordinates, we relate integration over S and integration over Ω using the pull-back

$$\int_S |\nabla_g \tilde{f}| \, \mathrm{d}s = \int_\Omega |\nabla_g \tilde{f}| \sqrt{G} \, \mathrm{d}x \mathrm{d}y, \tag{3.46}$$



(a) Flat surface **(b)** Ramp surface **(c)** Sine surface

Figure 3.11: ROF denoising on different manifolds. A flat surface (a) gives the same result as standard ROF denoising, but more complicated surfaces (b)(c) significantly change the result. The graph function $t(x, y)$ is depicted in the upper right corner. We can see that a ramp surface (b) produces regularization anisotropy due to the fact that the surface gradient is zero in y-direction but non-zero in x-direction. The same is true for the sine surface (c), where we can see strong regularization along level sets of the surface and less regularization across level sets.

where $\sqrt{G}$ is the differential area element that links distortion of the surface element $\mathrm{d}s$ to local coordinates $\mathrm{d}x\mathrm{d}y$. In the same spirit, we can pull back the data term defined on the manifold to the local coordinate domain Ω. In contrast to the minimal-area stereo regularizer in section 3.1, where we used the differential area element as regularization term, here we formulate the full variational model on the manifold, thus incorporating spatial as well as temporal information.

To assess the effect of TV_g as a regularization term, we depict in fig. 3.11 results of the following variant of the ROF denoising model [Rudin et al., 1992]

$$\min_u \int_\Omega |\nabla_g u| \sqrt{G} + \tfrac{\lambda}{2} |u - f|^2 \sqrt{G}\, \mathrm{d}x\mathrm{d}y, \tag{3.47}$$

with different $t(x, y)$, i.e. ROF-denoising on different manifolds. We see that computing the TV norm on the manifold can be interpreted as introducing anisotropy based on the surface geometry (see figs. 3.11(b) and 3.11(c)). We will use this to guide regularization of the reconstructed image according to the surface defined by the event time.

Data Term The data term $D(u, f^n)$ encodes the deviation of u from the noisy measurement f^n eq. (3.37). Under the reasonable assumption that a neuromorphic camera sensor suffers from the same noise as a conventional sensor, the measured update caused by one event will contain noise. In computer vision, a widespread approach is to model image noise as zero-mean additive Gaussian. While this simple model is sufficient for many applications, real sensor noise is dependent on scene brightness and should be modelled as a Poisson distribution [Ratner and Schechner, 2007]. We therefore define our data term as

$$D(u, f^n) := \lambda \int_S (u - f^n \log u)\, \mathrm{d}s = \lambda \int_\Omega (u - f^n \log u) \sqrt{G}\, \mathrm{d}x\mathrm{d}y \tag{3.48}$$
$$\text{s.t. } u(x, y) \in [u_{\min}, u_{\max}]$$

whose minimizer is known to be the correct ML-estimate under the assumption of Poisson-distributed noise between u and f^n [Le et al., 2007]. Note that in contrast to the minimal surface stereo model section 3.1, we also define the data term to lie on the manifold. eq. (3.48) is also known as *generalized Kullback-Leibler divergence* and has been investigated by [Steidl and Teuber, 2010] in variational image restoration methods. Furthermore, the data term is convex, which makes it easy to incorporate into our variational energy minimization framework. We restrict the range of $u(x, y) \in [u_{\min}, u_{\max}]$ since our reconstruction problem is defined up to a gray value offset caused by the unknown initial image intensities.

Discrete Energy In the discrete setting, we represent images of size $M \times M$ as matrices in $\mathbb{R}^{M\times M}$ with indices $(i, j) = 1 \dots M$. Derivatives are represented as linear maps $L_x, L_y : \mathbb{R}^{M\times M} \to \mathbb{R}^{M\times M}$, which are simple first order finite difference approximations of the derivative in x- and y-direction [Chambolle, 2004]. The discrete version of ∇_g, defined in eq. (3.44), can then be represented as a linear map $L_g : \mathbb{R}^{M\times M} \to \mathbb{R}^{M\times M\times 3}$ that acts on u as follows

$$
\begin{aligned}
(L_g u)_{ij1} &= \tfrac{1}{G_{ij}} \big((1 + (L_y t)^2_{ij})(L_x u)_{ij} - (L_x t)_{ij}(L_y t)_{ij}(L_y u)_{ij}\big) \\
(L_g u)_{ij2} &= \tfrac{1}{G_{ij}} \big((1 + (L_x t)^2_{ij})(L_y u)_{ij} - (L_x t)_{ij}(L_y t)_{ij}(L_x u)_{ij}\big) \\
(L_g u)_{ij3} &= \tfrac{1}{G_{ij}} \left((L_x t)_{ij}(L_x u)_{ij} + (L_y t)_{ij}(L_y u)_{ij}\right)
\end{aligned}
$$

Here, $G \in \mathbb{R}^{M\times M}$ is the pixel-wise determinant of the metric tensor given by $G_{ij} = 1 + (L_x t)^2_{ij} + (L_y t)^2_{ij}$.[4] The discrete data term follows from eq. (3.48) as $D(u, f^n) := \lambda \sum_{i,j}(u_{ij} - f^n_{ij} \log u_{ij})\sqrt{G_{ij}}$. This yields the complete discrete energy

$$
\min_u \|L_g u\|_g + \lambda \sum_{i,j} \left(u_{ij} - f^n_{ij} \log u_{ij}\right) \sqrt{G_{ij}} \quad \text{s.t. } u_{ij} \in [u_{\min}, u_{\max}], \tag{3.50}
$$

with the g-tensor norm defined as $\|A\|_g = \sum_{i,j} \sqrt{G_{ij} \sum_l (A_{ijl})^2} \quad \forall A \in \mathbb{R}^{M\times M\times 3}$.

3.2.2.2 Minimizing the Energy

We minimize eq. (3.50) using the Primal-Dual algorithm [Chambolle and Pock, 2011]. Dualizing the g-tensor norm yields the primal-dual formulation

$$
\min_u \max_p \left[D(u, f^n) + \langle L_g u, p\rangle - R^*(p)\right], \tag{3.51}
$$

where $u \in \mathbb{R}^{M\times M}$ is the discrete image, $p \in \mathbb{R}^{M\times M\times 3}$ is the dual variable and R^* denotes the convex conjugate of the g-tensor norm. A solution of eq. (3.51) is obtained by iterating

[4] The discrete element G_{ij}, i.e. the value of G at pixel position (ij) should not be confused with the metric tensor g_{ij}.

$$
\begin{aligned}
u_{k+1} &= \operatorname{prox}_{\tau D}(u_k - \tau L_g^* p_k) \\
p_{k+1} &= \operatorname{prox}_{\sigma R^*}(p_k + \sigma L_g(2u_{k+1} - u_k)),
\end{aligned}
$$

where L_g^* denotes the adjoint operator of L_g. The proximal maps for the data term and the regularization term can be solved in closed form, leading to the following update rules

$$
\begin{aligned}
\hat{u} = \operatorname{prox}_{\tau D}(\bar{u}) \qquad &\Leftrightarrow \hat{u}_{ij} = \operatorname*{clamp}_{u_{\min}, u_{\max}} \left(\tfrac{1}{2} \left(\bar{u}_{ij} - \beta_{ij} + \sqrt{(\bar{u}_{ij} - \beta_{ij})^2 + 4\beta_{ij} f_{ij}^n} \right) \right) \\
\hat{p} = \operatorname{prox}_{\sigma R^*}(\bar{p}) \qquad &\Leftrightarrow \hat{p}_{ijl} = \frac{\bar{p}_{ijl}}{\max\{1, \|\bar{p}_{ij,\cdot}\| / \sqrt{G_{ij}}\}},
\end{aligned}
$$

with $\beta_{ij} = \tau\lambda\sqrt{G_{ij}}$. The time-steps τ, σ are set according to $\tau\sigma \leq 1/\|L_g\|^2$, where we estimate the operator norm as $\|L_g\|^2 \leq 8 + 4\sqrt{2}$. Since the updates are pixel-wise independent, the algorithm can be efficiently parallelized on *GPUs*. Moreover, due to the low number of events added in each step, the algorithm usually converges in $k \leq 50$ iterations.

3.2.3 Experiments

We perform our experiments using a DVS128 camera with a spatial resolution of 128×128 and a temporal resolution of $1\,\mu s$. The parameter λ is kept fixed for all experiments. The thresholds Δ^+, Δ^- are set according to the chosen camera settings. In practice, the timestamps of the recorded events cannot be used directly as the manifold defined in section 3.2.2.1 due to noise. We therefore denoise the timestamps with a few iterations of a TV-L1 denoising method. We compare our method to the recently proposed method of [Bardow et al., 2016] on sequences provided by the authors. Furthermore, we will show the influence of the proposed regularization on the event manifold using a few self-recorded sequences.

3.2.3.1 Timing

In this work we aim for a real-time reconstruction method. We implemented the proposed method in C++ and used a Linux computer with a 3.4 GHz processor and a NVidia Titan X *GPU*[5]. Using this setup we measure a wall clock time of **1.7 ms** to create one single image, which amounts to $\approx$ 580 fps. While we can create a new image for each new event, this would create a tremendous amount of images due to the number of events ($\approx$ 500.000 per second on natural scenes with moderate camera movement). Furthermore one is limited by the monitor refresh rate of 60 Hz to actually display the images. In order to achieve real-time performance, one has two parameters: the number of events that are

[5]We note that the small image size of 128×128 is not enough to fully load the *GPU* such that we measured almost the same wall clock time on a NVidia 780 GTX Ti.

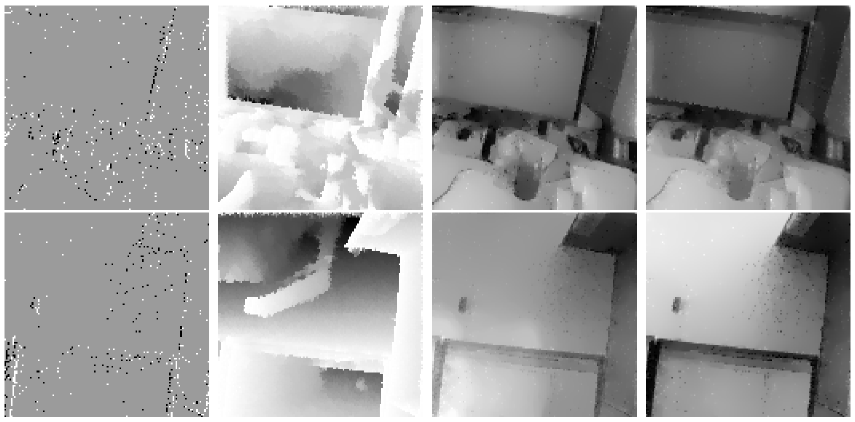


Figure 3.12: Sample results from our method. The columns depict (a) raw events, (b) time manifold, (c) result without manifold regularization and finally (d) with our manifold regularization. Notice the increased contrast in weakly textured regions, especially around the edge of the monitor.

integrated into one image and the number of frames skipped for display on screen. The results in the following sections have been achieved by accumulating 500 events to produce one image, which amounts to a time resolution of 3-5 ms.

3.2.3.2 Influence of the Event Manifold

We have captured a few sequences around our office with a DVS128 camera. In fig. 3.12 we show a few reconstructed images as well as the raw input events and the time manifold. For comparison, we switched off the manifold regularization (by setting $t(x, y) = const$), which results in images with notably less contrast.

3.2.3.3 Comparison to Related Methods

In this section we compare our reconstruction method to the method proposed [Bardow et al., 2016]. The authors kindly provided us with the recorded raw events, as well as intensity image reconstructions at regular timestamps $\delta t = 15$ms. Since we process shorter event packets, we search for the nearest neighbour timestamp for each image of [Bardow et al., 2016] in our sequences. We visually compare our method on the sequences *face*, *jumping jack* and *ball* to the results of [Bardow et al., 2016]. We point out that no ground truth data is available so we are limited to purely qualitative comparisons.

In fig. 3.13 we show a few images from the sequences. Since we are dealing with highly dynamic data, we point the reader to the included supplementary video[6] which shows whole sequences of several hundred frames.

[6]`https://www.youtube.com/watch?v=rvB2URrGT94`

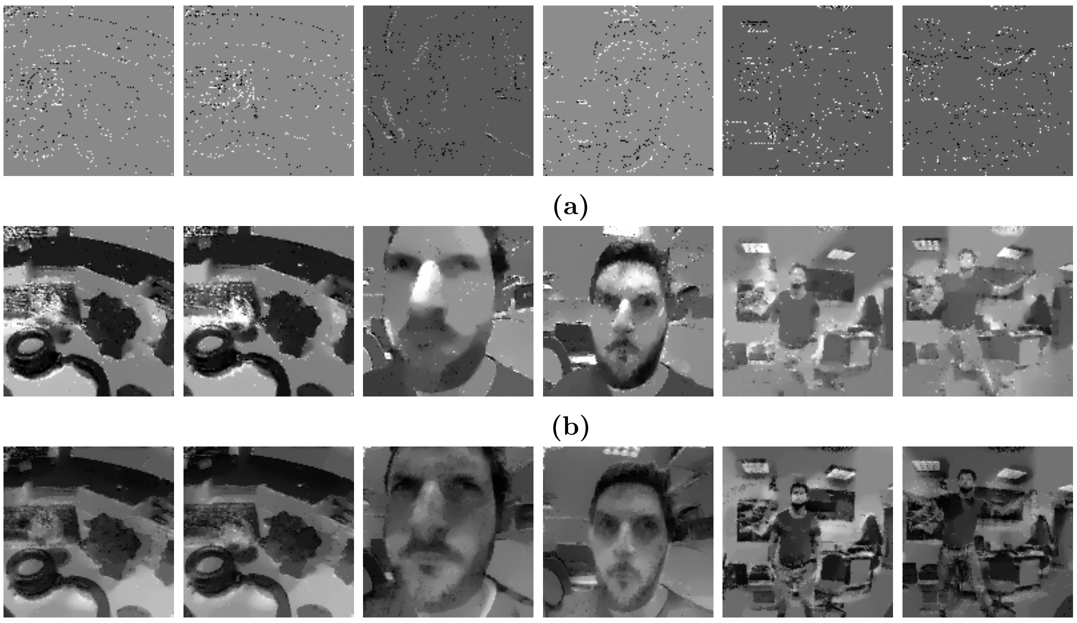


Figure 3.13: Comparison to the method of [Bardow et al., 2016]. The first row (a) shows the raw input events that have been used for both methods. The second row (b) depicts the results of Bardow et al., and the last row (c) shows our result. We can see that out method produces more details (e.g. face, beard) as well as more graceful gray value variations in untextured areas, where [Bardow et al., 2016] tends to produce a single gray value.

3.2.3.4 Comparison to Standard Cameras

We have captured a sequence using a DVS128 camera as well as a Canon EOS60D Digital Single-Lens Reflex (DSLR) camera to compare the fundamental differences of traditional cameras and event-based cameras. As already pointed out by [Bardow et al., 2016], rapid movement results in motion blur for conventional cameras, while event-based cameras show no such effects. Also the dynamic range of a *DVS* is much higher, which is also shown in fig. 3.14.

3.2.4 Conclusion

In this paper we have proposed a method to recover intensity images from neuromorphic or event cameras in real-time. We cast this problem as an iterative filtering of incoming events in a variational denoising framework. We propose to utilise a manifold that is induced by the timestamps of the events to guide the image restoration process. This allows us to incorporate information about the relative ordering of incoming pixel information without explicitly estimating optical flow like in previous works. This in turn enables an efficient algorithm that can run in real-time on currently available PCs.

Future work will include the study of the proper noise characteristic of event cameras.

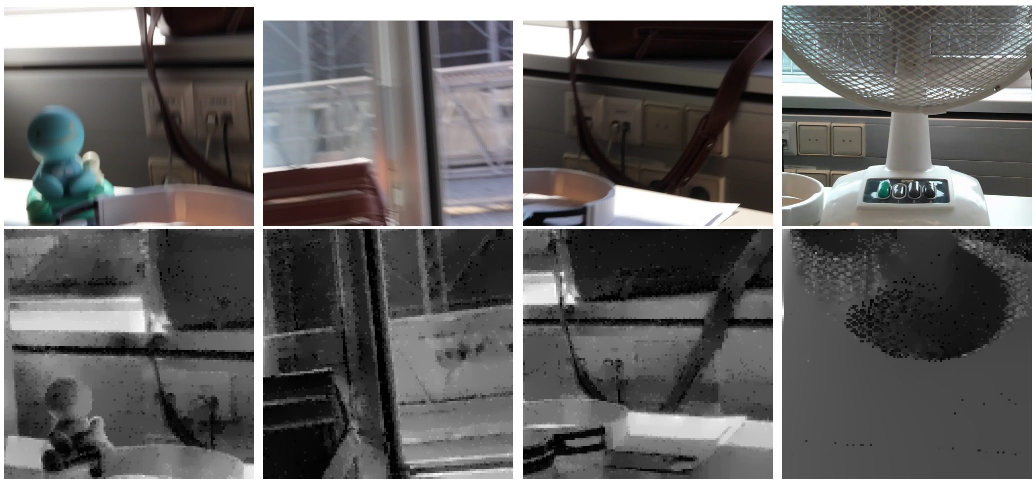


Figure 3.14: Comparison to a video captured with a modern *DSLR* camera. Notice the rather strong motion blur in the images of the *DSLR* (top row), whereas the *DVS* camera can easily deal with fast camera or object movement (bottom row).

While the current model produces natural-looking intensity images, a few noisy pixels appear that indicate a still non-optimal treatment of sensor noise within our framework. Also it might be beneficial to look into a local minimization of the energy on the manifold (e.g. by coordinate-descent) to further increase the processing speed.

4 Learning Descriptors for Optical Flow

Contents

Optical flow can be seen as an instance of the dense image matching problem, where the goal is to find for each pixel its corresponding match in the other image. One fundamental question in the dense matching problem is how to choose good *descriptors* or *features*. Data mining with Convolutional Neural Networks (CNNs) has recently shown excellent results for learning task-specific image features, outperforming previous methods based on hand-crafted descriptors. One of the major difficulties in learning features for optical flow is the high dimensionality of the cost function: Whereas in stereo, the full cost function can be represented as a 3D volume, the matching cost in optical flow is a 4D volume. Especially at high image resolutions, operations on the flow matching cost are expensive both in terms of memory requirements and computation time.

Our method avoids explicit storage of the full cost volume, both in the learning phase and during inference. This is achieved by a *splitting* (or *min-projection*) of the 4D cost into two quasi-independent 3D volumes, corresponding to the u and v component of the flow. We then formulate *CNN* learning and Conditional Random Field (CRF) inference in this reduced setting. This achieves a space complexity linear in the size of the search range, similar to recent stereo methods, which is a significant reduction compared to the quadratic complexity of the full 4D cost function.

Nevertheless, we still have to compute all entries of the 4D cost function. This computational bottleneck can be optimized by using binary descriptors, which give a theoretical speed-up factor of 32. In practice, even larger speed-up factors are attained, since binary descriptors need less memory bandwidth and also yield a better cache efficiency. Consequently, we aim to incorporate a binarization step into the learning. We propose a novel hybrid learning scheme, where we circumvent the problem of hard nonlinearities having zero gradient. We show that our hybrid learning performs almost as well as a network without hard nonlinearities, and much better than the previous state of the art in learning binary *CNNs*.

4.1 Related Work

In the past hand-crafted descriptors like Scale-Invariant Feature Transform (SIFT), Normalized Cross-Correlation (NCC), Features from Accelerated Segment Test (FAST) etc. have been used extensively with very good results, but recently *CNN*-based approaches [Žbontar and LeCun, 2015, Luo et al., 2016] marked a paradigm shift in the field of image matching. To date all top performing methods in the major stereo benchmarks rely heavily on features learned by *CNNs*. For optical flow, many recent works still use engineered features [Chen and Koltun, 2016, Bailer et al., 2015], presumably due to the difficulties the high dimensional optical flow cost function poses for learning. Only very recently we see a shift towards *CNNs* for learning descriptors [Gadot and Wolf, 2016, Güney and Geiger, 2016, Xu et al., 2017]. Our work is most related to [Xu et al., 2017], who construct the full 4D cost volume and run an adapted version of Semiglobal Matching (SGM) on it. They perform learning and cost volume optimization on $\frac{1}{3}$ of the original resolution and compress the cost function in order to cope with the high memory consumption. Our method is memory-efficient thanks to the dimensionality-reduction by the min-projection, and we outperform the reported runtime of [Xu et al., 2017] by a factor of 10.

Full flow with *CRF* [Chen and Koltun, 2016] is a related inference method using Sequential Tree-Reweighted Message Passing (TRW-S) [Kolmogorov, 2006] with efficient distance transform [Felzenszwalb and Huttenlocher, 2006]. Its iterations have quadratic time and space complexity. In practice, this takes 20GB[1] of memory, and 10-30 sec. per iteration with a parallel CPU implementation. We use the decomposed model [Shekhovtsov et al., 2008] with a better memory complexity and a faster parallel inference scheme based on [Shekhovtsov et al., 2016].

Hand-crafted binary descriptors like Census have been shown to work well in a number of applications, including image matching for stereo and flow [Ranftl et al., 2014, Ranftl et al., 2012, Trzcinski et al., 2013, Calonder et al., 2010]. However, direct learning of binary descriptors is a difficult task, since the hard thresholding function, $\text{sign}(x)$, has gradient zero almost everywhere. In the context of Binary *CNNs* there are several approaches to train networks with binary activations [Bengio et al., 2013] and even binary weights [Courbariaux and Bengio, 2016, Rastegari et al., 2016]. This is known to give a considerable compression and speed-up at the price of a tolerable loss of accuracy. To

[1]Estimated for the cost volume size $341\times145\times160\times160$ based on numbers in [Chen and Koltun, 2016] corresponding to $\frac{1}{3}$ resolution of Sintel images.

circumvent the problem of $\mathrm{sign}(x)$ having zero gradient a.e., surrogate gradients are used. The simplest method, called *straight-through estimator* [Bengio et al., 2013] is to assume the derivative of $\mathrm{sign}(x)$ is 1, i.e., simply omit the sign function in the gradient computation. This approach can be considered as the state of the art, as it gives best results in [Bengio et al., 2013, Courbariaux and Bengio, 2016, Rastegari et al., 2016]. We show that in the context of learning binary descriptors for the purpose of matching, alternative strategies are possible which give better results.

4.2 Method

We define two models for optical flow: a local model, known as Winner-Takes-All (WTA) and a joint model, which uses *CRF* inference. Both models use *CNN* descriptors, learned in section 4.2.1.2. The joint model has only few extra parameters that are fit separately and the inference is solved with a parallel method, see section 4.2.2. For *CNN* learning, we optimize the performance of the local model. While learning by optimizing the performance of the joint model is possible [Knöbelreiter et al., 2017], the resulting procedures are significantly more difficult.

We assume color images $I^1, I^2 : \Omega \to \mathbb{R}^3$, where $\Omega = \{1, \dots H\} \times \{1, \dots W\}$ is a set of pixels. Let $\mathcal{W} = \mathcal{S} \times \mathcal{S}$ be a window of discrete 2D displacements, with $\mathcal{S} = \{-D/2, -D/2+1, \dots, D/2-1\}$ given by the search window size D, an even number. The flow $x : \Omega \to \mathcal{W}$ associates a displacement to each pixel $i \in \Omega$ so that the displaced position of i is given by $i + x_i \in \mathbb{Z}^2$. For convenience, we denote by $x = (u, v)$, where u and v are mappings $\Omega \to \mathcal{S}$, the components of the flow in horizontal and vertical directions, respectively. The per-pixel *descriptors* $\phi(I;\theta) : \Omega \to \mathbb{R}^m$ are computed by a *CNN* with parameters θ. Let ϕ^1, ϕ^2 be descriptors of images I^1, I^2, respectively. The *local matching cost* for a pixel $i \in \Omega$ and displacement $x_i \in \mathcal{W}$ is given by

$$c_i(x_i) = \begin{cases} d(\phi^1_i, \phi^2_{i+x_i}) & \text{if } i + x_i \in \Omega, \\ c_{\text{outside}} & \text{otherwise,} \end{cases} \tag{4.1}$$

where $d : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ is a distance function in $\mathbb{R}^m$. "Distance" is used in a loose sense here, we will consider the negative[2] scalar product $d(\phi^1, \phi^2) = -\langle \phi^1, \phi^2 \rangle$. We call

$$\hat{x}_i \in \arg\min_{x_i \in \mathcal{W}} c_i(x_i) \tag{4.2}$$

the *local optical flow model*, which finds independently for each pixel i a displacement x_i that optimizes the local matching cost. The *joint optical flow model* finds the full flow

[2]since we want to pose matching as a minimization problem

field x optimizing the coupled CRF energy cost:

$$\hat{x} \in \arg \min_{u,v\colon \Omega \to \mathcal{S}} \Big[\sum_{i \in \Omega} c_i(u_i, v_i) + \sum_{i \sim j} w_{ij}(\rho(u_i - u_j) + \rho(v_i - v_j)) \Big], \tag{4.3}$$

where $i \sim j$ denotes a 4-connected pixel neighborhood, w_{ij} are contrast-sensitive weights, given by $w_{ij} = \exp(-\frac{\alpha}{3} \sum_{c \in \{R,G,B\}} |I^1_{i,c} - I^1_{j,c}|)$ and $\rho\colon \mathbb{R} \to \mathbb{R}$ is a robust penalty function shown in fig. 4.1.

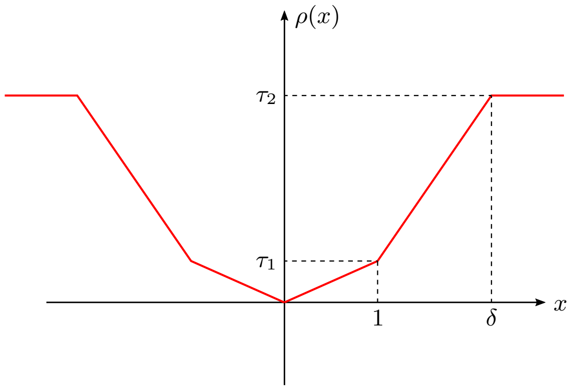


Figure 4.1: The penalty function ρ used by the *CRF* is a generalization of the $P1P2$ penalty function of *SGM* [Hirschmüller, 2005]. The cutoff is parameterized by δ, whereas in the $P1P2$ loss it is fixed at $|x| = 2$.

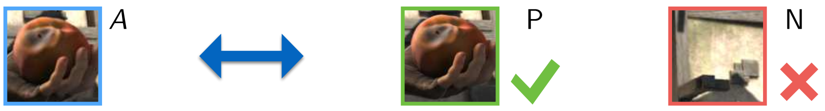
4.2.1 Learning Descriptors

We can identify two popular approaches to learning features for image matching. The first is based on a binary classification model. Assume we have given a descriptor $\phi^1 = \phi^1_i$ at some position i in the first image. With the groundtruth displacement x^{gt}_i, one obtains the positive example $\phi^{2,P} = \phi^2_{i+x^{\text{gt}}_i}$, the descriptor in the second image at position $i + x^{\text{gt}}_i$. Furthermore a negative example $\phi^{2,N} = \phi^2_{i+x^{\text{N}}_i}$ is sampled from some position $i+x^{\text{N}}_i$, where $x^{\text{N}}_i \neq x^{\text{gt}}_i$. Then the goal is to learn the descriptors such that

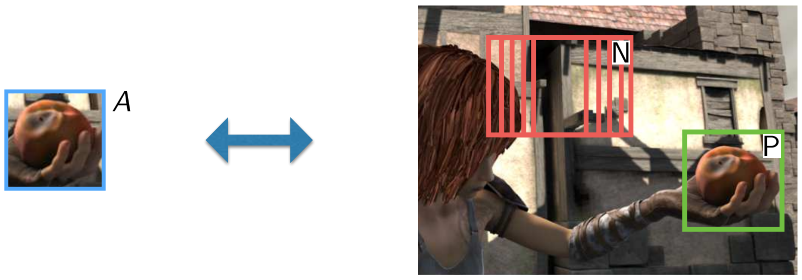
$$d(\phi^1, \phi^{2,P}) < d(\phi^1, \phi^{2,N}), \tag{4.4}$$

the distance between the true matching pair of descriptors should be smaller than the distance between the non-matching pair. In this sense it is a “one-vs-one” strategy, since each positive example competes against one negative example, see fig. 4.2(a). It is clear that this strategy requires a sampling strategy to obtain the negative samples [Simo-Serra et al., 2015]. In the context of learning features for optical flow, the sampling heuristic plays an important role and can potentially influence the learning results due to the large number of possibilities in choosing a negative example.

One the other hand, one could adopt a multiclass classification approach. In this setting



(a) Binary (one-vs-one) classification: Patch A in the first image is compared to the correct example P and to a single negative example N in the second image.



(b) Multiclass (one-vs-all) classification: Patch A in the first image is compared exhaustively to *all* examples in the second image. The classifier delivers a probability for all possible classes, i.e. displacements.

Figure 4.2: Binary (a) and multiclass (b) classification for *CNN*-learning.

one compares exhaustively against all negative examples $\phi_n^{2,N} = \phi^2_{i+x_i^n}$. Here, $x_i^n \in \mathcal{W}$ denotes all possible displacements in the search window $\mathcal{W}$ enumerated by $n = 1 \dots D^2$. Of course we need to exclude the correct displacement, hence we require that $x_i^n \neq x_i^{\text{gt}}\ \forall n \in \mathbb{N}$. This yields $n = D^2 - 1$ negative examples. The goal is now to learn the descriptors such that

$$d(\phi^1, \phi^{2,P}) < \min_n d(\phi^1, \phi_n^{2,N}), \tag{4.5}$$

the distance between the correct pair of descriptors should be smaller than the distance between all other pairs. In this sense it is a "one-vs-all" strategy, see fig. 4.2(b). In the context of stereo matching it has been shown that such approach yields good results [Luo et al., 2016]. The one-vs-all comparison gives the network additional information and allows to implicitly capture correlations between the different displacements. Note that the multiclass approach does not need a patch sampling step because all potential matches are considered at once.

In this work, we wish to apply the multiclass strategy eq. (4.5). We point out that in contrast to the binary classification model eq. (4.4), multiclass learning needs the full cost function eq. (4.1) in memory. In contrast to the stereo setting, storing the optical flow cost function needs $\mathcal{O}(|\Omega|D^2)$ memory, which is prohibitive at high image resolutions. To the best of our knowledge, all previous works that learned descriptors for optical flow exclusively used the binary classification approach [Xu et al., 2017, Güney and Geiger, 2016, Gadot and Wolf, 2016]. In the following we will show how to efficiently apply the multiclass approach at linear space complexity, which facilitates memory-efficient end-to-end training on high resolution images without a patch sampling step.

4.2.1.1 Dimensionality Reduction via Min-Projection

Processing the 4D cost eq. (4.1) involves computing distances in $\mathbb{R}^m$ per entry. Storing such cost volume takes $\mathcal{O}(|\Omega|D^2)$ space and evaluating it $\mathcal{O}(|\Omega|D^2m)$ time. We can reduce space complexity to $\mathcal{O}(|\Omega|D)$ by avoiding explicit storage of the 4D cost function. Towards this end we write the local optical flow model eq. (4.2) in the following way

$$\hat{u}_i \in \arg\min_{u_i} c_i^{\mathrm{U}}(u_i), \quad \text{where} \quad c_i^{\mathrm{U}}(u_i) = \min_{v_i} c_i(u_i, v_i); \tag{4.6a}$$

$$\hat{v}_i \in \arg\min_{v_i} c_i^{\mathrm{V}}(v_i), \quad \text{where} \quad c_i^{\mathrm{V}}(v_i) = \min_{u_i} c_i(u_i, v_i). \tag{4.6b}$$

The inner step in (4.6a) and (4.6b), called *min-projection*, minimizes out one component of the flow vector. This can be interpreted as a decoupling of the full 4D flow problem into two simpler quasi-independent 3D problems on the reduced cost volumes $c^{\mathrm{U}}, c^{\mathrm{V}}$, see fig. 4.3. Assuming the minimizer of (4.2) is unique, (4.6a) and (4.6b) find the same solution as the original problem (4.2). Using this representation, *CNN* learning can be implemented within existing frameworks. We point out that this approach has the same space complexity as recent methods for learning stereo matching, since we only need to store the 3D cost volumes c^{U} and c^{V}. As an illustrative example consider an image with size 1024×436 and a search range of 256. In this setting the full 4D cost function takes roughly 108 GB whereas our splitting consumes only 0.8 GB.

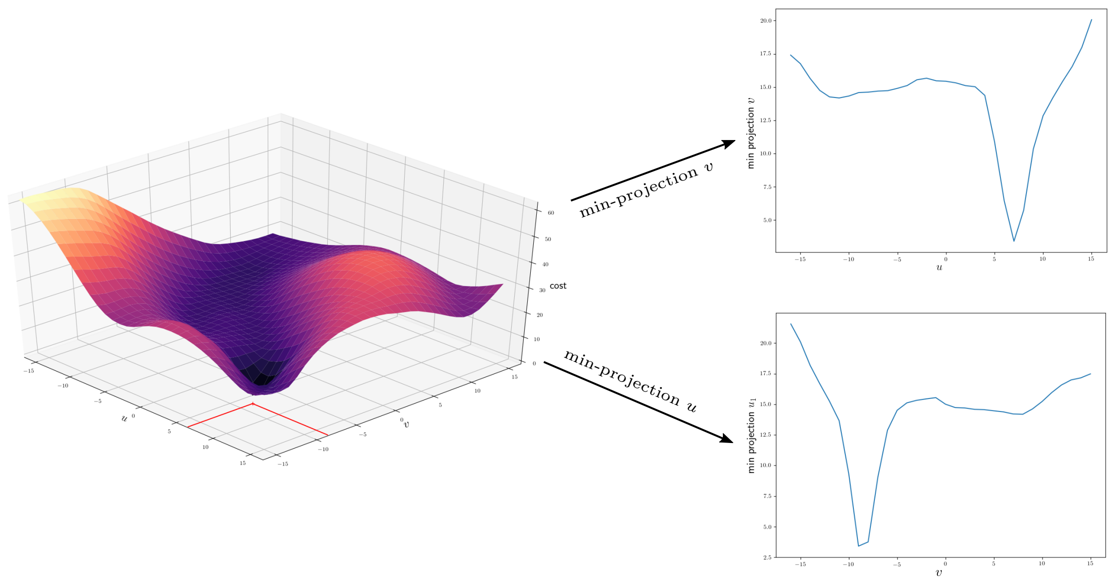


Figure 4.3: Solving the local flow model eq. (4.2) for a fixed pixel i amounts to minimizing a 2D cost function depicted on the left side. The minimum, marked in red, can be alternatively computed by considering the min-projections along the u and v dimension of the flow, depicted on the right side. The advantage is that the min-projections are one-dimensional functions, requiring only $\mathcal{O}(|\Omega|D)$ memory to store.

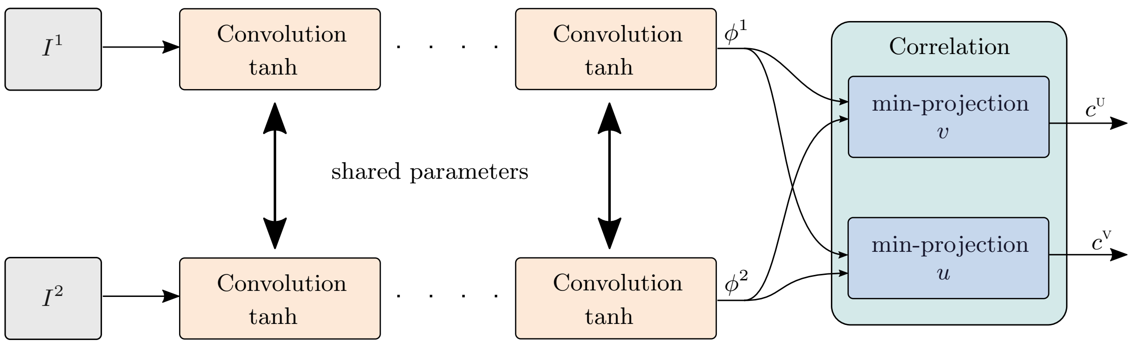


Figure 4.4: Network architecture: A number of convolutional layers with shared parameters computes feature vectors ϕ^1, ϕ^2 for every pixel. These feature vectors are cross-fed into a correlation layer, that computes local matching costs in u and v direction by minimizing out the other direction. The result are two quasi-independent cost volumes for the u and v component of the flow.

4.2.1.2 Network

Figure 4.4 shows the network diagram of the local flow model eq. (4.2). The structure is similar to the recent methods proposed for learning stereo matching [Luo et al., 2016, Žbontar and LeCun, 2015, Chen et al., 2015, Knöbelreiter et al., 2017]. It is a siamese network consisting of two convolutional branches with shared parameters, followed by a correlation layer. We use a filter size of 3×3 for the convolutions. The tanh nonlinearity keeps feature values in a defined range, which works well with the scalar product as distance function. We do not use striding or pooling. The last convolutional layer uses 96 filter channels which fixes the dimensionality of the distance space to $m = 96$. The previous layers use a varying number of filter channels, where we found that starting with a small number (e.g. 16 in the first layer) and gradually increasing to 96 in the last layer yields good results.

Loss Given the groundtruth flow field (u^*, v^*), we pose the learning objective as follows: we define a probabilistic softmax model of the local prediction u_i (resp. v_i) as $p(u_i) \propto \exp(-sc^{\mathrm{U}}_i(u_i))$, where $s \in \mathbb{R}$ is a global scale factor learned by the network. This gives the network the freedom to adapt for a more smooth resp. peaky distribution $p(u_i)$. Then we consider a naive model $p(u, v) = \prod_i p(u_i)p(v_i)$ and apply the maximum likelihood criterion. The negative log likelihood is given by

$$L(u, v) = -\sum_{i\in\Omega} \big[\log p(u^*_i) + \log p(v^*_i)\big]. \tag{4.7}$$

This is equivalent to cross-entropy loss with the target distribution concentrated at the single point (u^*_i, v^*_i) for each i. Variants of the cross-entropy loss, where the target distribution is spread around the ground truth point (u^*_i, v^*_i) are also used in the literature [Luo et al., 2016] and can be easily incorporated.

4.2.1.3 Learning Quantized Descriptors

The computational bottleneck in scheme (4.6) is computing the min-projections, with time complexity $\mathcal{O}(|\Omega|D^2 m)$. This operation arises during the learning as well as in the *CRF* inference step, where it corresponds to the message exchange in the dual decomposition. It is therefore desirable to accelerate this step. We achieve a significant speed-up by quantizing the descriptors and evaluating the Hamming distance of binary descriptors.

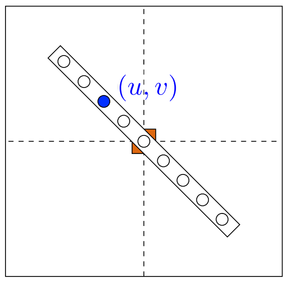
Let us define the quantization: we call $\bar{\phi} = \text{sign}(\phi)$ the *quantized descriptor field*. The distance between quantized descriptors is given by $d(\bar{\phi}^1, \bar{\phi}^2) = -\langle \bar{\phi}^1, \bar{\phi}^2 \rangle = 2\mathcal{H}(\bar{\phi}^1, \bar{\phi}^2) - m$, equivalent to the Hamming distance $\mathcal{H}(\cdot, \cdot)$ up to a scaling and an offset. Let the quantized cost function be denoted $\bar{c}_i(x_i)$, defined similar to (4.1). We can then compute quantized min-projections $\bar{c}^{\text{U}}$, $\bar{c}^{\text{V}}$.

However, learning model (4.2) with quantized descriptors is difficult due to the gradient of the sign function being zero almost everywhere. We introduce a new technique specific to the matching problem and compare it to the baseline method that uses the straight-through estimator of the gradient [Bengio et al., 2013]. Consider the following variants of the model (4.6a)

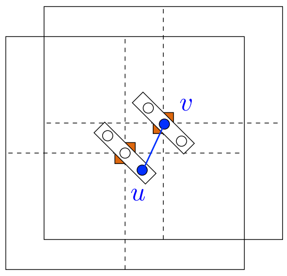
$$\hat{u}_i \in \arg\min_{u_i} c_i(u_i, \hat{v}_i(u_i)), \quad \text{where} \quad \hat{v}_i(u_i) \in \arg\min_{v_i} \bar{c}_i(u_i, v_i); \tag{FQ}$$

$$\hat{u}_i \in \arg\min_{u_i} \bar{c}_i(u_i, \hat{v}_i(u_i)), \quad \text{where} \quad \hat{v}_i(u_i) \in \arg\min_{v_i} \bar{c}_i(u_i, v_i). \tag{QQ}$$

The respective variants of (4.6b) are symmetric. The second letter in the naming scheme indicates whether the inner problem, i.e., the min-projection step, is performed on (Q)uantized or (F)ull cost, whereas the first letter refers to the outer problem on the smaller 3D cost volume. The initial model (4.6a) is thus also denoted as FF model. While models FF and QQ correspond, up to non-uniqueness of solutions, to the joint minimum in (u_i, v_i) of the cost c and $\bar{c}$ respectively, the model FQ is a mixed one. This hybrid model is interesting because minimization in v_i can be computed efficiently on the binarized cost with Hamming distance, and the minimization in u_i has a non-zero gradient in c^{U}. We thus consider the model FQ as an efficient variant of the local optical flow model (4.2). In addition, it is a good learning proxy for the model QQ: Let $\hat{u}_i = \arg\min_{u_i} c_i(u_i, \hat{v}_i(u_i))$ be a minimizer of the outer problem FQ. Then the derivative of FQ is defined by the indicator of the pair $(\hat{u}_i, \hat{v}_i(u_i))$. This is the same as the derivative of FF, except that $\hat{v}_i(u_i)$ is computed differently. Learning the model QQ involves a hard quantization step, and we apply the straight-through estimator to compute a gradient. Note that the exact gradient for the model FQ can be computed at approximately the same reduced computational cost as the straight-through gradient in the model QQ.



(a) The labels of the product model are two-dimensional displacements. The graph consists of $|\Omega|$ variables and D^2 labels, the matching cost lives on the labels.



(b) The labels of the decomposed model are one-dimensional displacements. The graph consists of $2|\Omega|$ variables and D labels, the matching cost lives on the inter-layer edges.

Figure 4.5: *CRF* product model (a) and decomposed model (b).

4.2.2 CRF

The baseline model, which we call *product* model, has $|\Omega|$ variables x_i with the state space $\mathcal{S} \times \mathcal{S}$, see fig. 4.5(a). It has been observed in [Felzenszwalb and Huttenlocher, 2006] that max-product message passing in the *CRF* eq. (4.3) can be computed in time $\mathcal{O}(D^2)$ per variable for separable interactions using a fast distance transform. However, storing the messages for a 4-connected graph requires $\mathcal{O}(|\Omega|D^2)$ memory. Although such an approach was shown feasible even for large displacement optical flow [Chen and Koltun, 2016], we argue that a more compact *decomposed* model [Shekhovtsov et al., 2008] gives comparable results and is much faster in practice. The decomposed model is constructed by observing that the regularization in eq. (4.3) is separable over u and v. Then the energy eq. (4.3) can be represented as a *CRF* with $2|\Omega|$ variables u_i, v_i (see fig. 4.5(b)) with the following pairwise terms: The in-plane term $w_{ij}\rho(u_i - u_j)$ and the cross-plane term $c(u_i, v_i)$, forming the graph shown in fig. 4.6(a). In this formulation there are no unary terms, since costs c_i are interpreted as pairwise terms. The resulting Linear Programming (LP) dual is more economical, because it has only $\mathcal{O}(|\Omega|D)$ variables. The message passing for edges inside planes and across planes has complexity $\mathcal{O}(|\Omega|D)$ and $O(|\Omega|D^2)$, respectively. The price for the $\mathcal{O}(|\Omega|D)$ memory complexity of the decomposed model is that it admits a weaker *LP*-relaxation compared to the product model. We argue that it is a favorable trade, since solving the product model on high resolution images is simply intractable due to the immense memory requirements.

We apply the parallel inference method [Shekhovtsov et al., 2016] to the dual of the decomposed model [Shekhovtsov et al., 2008], see fig. 4.6(a). Although different dual decompositions reach different objective values in a fixed number of iterations, it is known that all decompositions with trees covering the graph are equivalent in the optimal value [Wainwright et al., 2005]. The decomposition in fig. 4.6(a) is into horizontal and vertical chains in each of the u- and v- planes plus a subproblem containing all cross-layer edges. We introduce Lagrange multipliers $\lambda = (\lambda^k \in \mathbb{R}^{\Omega \times \mathcal{S}} \mid k = 1, 2, 3, 4)$

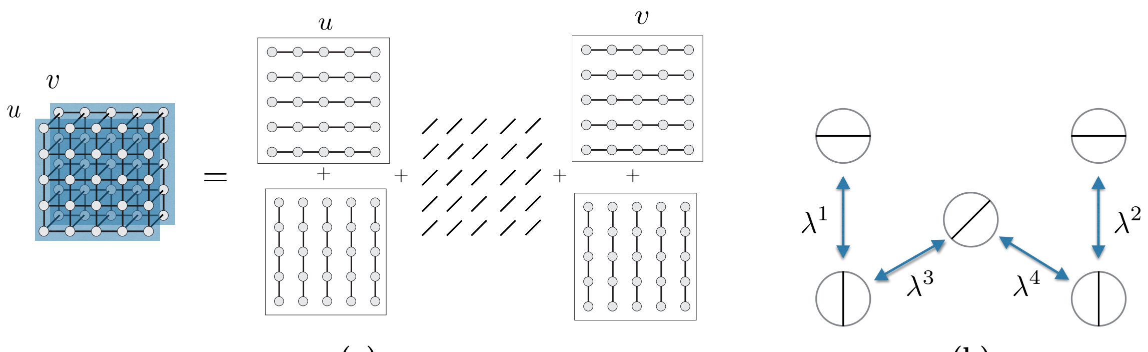


Figure 4.6: (a) Decomposition of the pairwise CRF into 5 subproblems. (b) Lagrange multipliers in the dual corresponding to equality constraints between the subproblems. They act as offsets of unary costs between subproblems, increasing on one side of the arrow and decreasing on the other.

enforcing equality constraints between the subproblems as shown in fig. 4.6(b). The Lagrange multipliers λ^k are identified with modular functions $\lambda^k\colon \mathcal{S}^\Omega \to \mathbb{R}\colon u \mapsto \sum_i \lambda^k_i(u_i)$. Let us also introduce shorthands for the sum of pairwise terms over horizontal chains $f^{\mathrm{h}}\colon \mathcal{S}^\Omega \to \mathbb{R}\colon u \mapsto \sum_{ij\in\mathcal{E}^{\mathrm{h}}} w_{ij}\rho(u_i - u_j)$, and a symmetric definition f^{v} for the sum over the vertical chains. The lower bound $\Psi(\lambda)$ corresponding to the decomposition in fig. 4.6(b) is given by:

$$\Psi(\lambda) = \Psi^1(\lambda) + \Psi^2(\lambda) + \Psi^3(\lambda), \text{ where} \tag{4.8a}$$

$$\Psi^1(\lambda) = \min_u \big[(\lambda^1 + \lambda^3)(u) + f^{\mathrm{h}}(u)\big] + \min_u \big[-\lambda^1(u) + f^{\mathrm{v}}(u)\big]; \tag{4.8b}$$

$$\Psi^2(\lambda) = \min_v \big[(\lambda^2 + \lambda^4)(v) + f^{\mathrm{h}}(v)\big] + \min_v \big[-\lambda^2(v) + f^{\mathrm{v}}(v)\big]; \tag{4.8c}$$

$$\Psi^3(\lambda) = \sum_i \min_{u_i,v_i} \big[c_i(u_i,v_i) - \lambda^3_i(u_i) - \lambda^4_i(v_i)\big]. \tag{4.8d}$$

Our Lagrangian dual to eq. (4.3) is to maximize $\Psi(\lambda)$ in λ, which enforces consistency between minimizers of the subproblems. The general theory [Wainwright et al., 2005] applies, in particular, when the minimizers of all subproblems are consistent they form a global minimizer. In eq. (4.8b), there is a sum of horizontal and vertical chain subproblems in the u-plane. When λ^3 is fixed, $\Psi^1(\lambda)$ is the lower bound corresponding to the relaxation of the energy in u with the unary terms given by λ^3. It can be interpreted as a stereo-like problem with 1D labels u. Similarly, $\Psi^2(\lambda)$ is a lower bound for the v-plane with unary terms λ^4. Subproblem $\Psi^3(\lambda)$ is simple, it contains both variables u, v but the minimization decouples over individual pairs (u_i, v_i). It connects the two stereo-like problems through the 4D cost volume c.

Updating messages inside planes can be done at a different rate than across planes. The optimal rate for fast convergence depends on the time complexity of the message updates. [Shekhovtsov et al., 2008] reported an optimal rate of updating in-plane messages 5 times as often using the *TRW-S* solver [Kolmogorov, 2006].

Algorithm 4.1: Flow CRF Optimization

```
  Input: Cost volume $c$;
  Output: Dual point $\lambda$ optimizing $\Psi(\lambda)$;
1 Initialize $\lambda := 0$;
2 for $t = 1, \dots,$ it_outer do
3     Perform the following updates:
4         $v \to u$: pass slacks to $u$-plane by eq. (4.11), changes $\lambda^3$;
5         $u$-plane: DMM with it_inner iterations for $u$-plane eq. (4.9), changes $\lambda^1$, $\lambda^3$ ;
6         $u \to v$: pass slacks to $v$-plane by eq. (4.10), changes $\lambda^4$;
7         $v$-plane: DMM with it_inner iterations for $v$-plane, changes $\lambda^2$, $\lambda^4$;
8 end
```

The decomposition eq. (4.8a) facilitates this kind of strategy and allows to use the implementation [Shekhovtsov et al., 2016] designed for stereo-like problems. We therefore use the dual solver [Shekhovtsov et al., 2016], denoted Dual Minorize-Maximize (DMM) to perform in-plane updates. When applied to the problem of maximizing $\Psi^1(\lambda)$ in λ^1, it has the following properties: a) the bound $\Psi^1(\lambda)$ does not decrease and b) it computes a modular minorant s such that $s(u) \leq \lambda^3(u) + f^{\mathrm{h}}(u) + f^{\mathrm{v}}(u)$ for all u and $\Psi^1(\lambda) = \sum_i \min_{u_i} s_i(u_i)$. The modular minorant s is an excess of costs, called *slacks*, which can be subtracted from λ^3 while keeping $\Psi^1(\lambda)$ non-negative. The associated update of the u-plane can be denoted as

$$\begin{aligned} (\lambda^1, s) &:= \mathrm{DMM}(\lambda^1, \lambda^3, f^{\mathrm{h}}, f^{\mathrm{v}}), \\ \lambda^3 &:= \lambda^3 - s. \end{aligned} \tag{4.9}$$

The slack s is then passed to the v plane by the following updates, i.e., message passing $u \to v$:

$$\lambda^4_i(v_i) := \lambda^4_i(v_i) + \min_{u_i} \big[c_i(u_i, v_i) - \lambda^3_i(u_i)\big]. \tag{4.10}$$

The minimization eq. (4.10) has time complexity $\mathcal{O}(|\Omega|D^2)$, assuming the 4D costs c_i are available in memory. As discussed above, we can compute the costs c_i efficiently on the fly and avoid $\mathcal{O}(|\Omega|D^2)$ storage. The update $v \to u$ is symmetric to eq. (4.10):

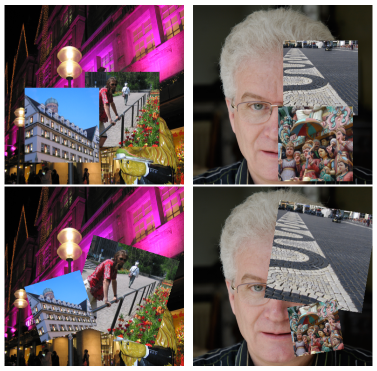
$$\lambda^3_i(u_i) := \lambda^3_i(u_i) + \min_{v_i} \big[c_i(u_i, v_i) - \lambda^4_i(v_i)\big]. \tag{4.11}$$

The complete method is summarized in algorithm 4.1. It starts from collecting the slacks in the u-plane. When initialized with $\lambda = 0$, the update eq. (4.11) simplifies to $\lambda^3_i(u_i) = \min_{v_u} c_i(u_i, v_i)$, i.e., it is exactly matching to the min-projection c^{U} eq. (4.6). The problem solved with *DMM* in line 5 in the first iteration is a stereo-like problem with cost c^{U}. The dual solution redistributes the costs and determines which values of u are worse than others, and expresses this cost offset in λ^3 as specified in eq. (4.9). The optimization of the v-plane then continues with some information of good solutions for u propagated via the cost offsets using eq. (4.10).

4.3 Evaluation

4.3.1 Learning Patch Transformations

(a) Isometry: Rotation and translation.


(b) Similarity: additional global scale factor.

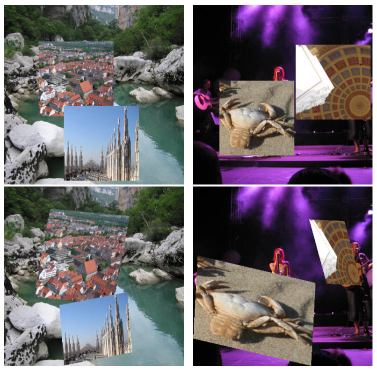

(c) Affine: additional non-isotropic scaling.

Figure 4.7: Sample training data for learning patch transformations.

We conduct experiments in order to evaluate the ability of the *CNN* to learn different patch transformations. To that end, we generated training data by applying different homographies to image patches. A homography $H \in \mathbb{R}^{3\times3}$ may be subclassed into

- Isometries $H = \begin{bmatrix} R(\theta) & t \\ 0^T & 1 \end{bmatrix}$ consisting of a rotation $R(\theta) \in \mathrm{SO}(2)$ where $R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ and a translation $t = (t_x, t_y)$. An isometry has 3 Degrees of Freedom (DOF): the rotation angle θ and the two translation parameters.

- Similarities $H = \begin{bmatrix} sR(\theta) & t \\ 0^T & 1 \end{bmatrix}$ are specified by an isometry with an additional scaling $s \in \mathbb{R}$. A similarity has 4 *DOF*.

- Affine transformations $H = \begin{bmatrix} A & t \\ 0^T & 1 \end{bmatrix}$, where $A = R(\theta)R(-\varphi)DR(\varphi)$ and $D = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$. Compared to the similarity, an affine transformation has two additional parameters: The shearing angle φ and the ratio of the scaling parameters λ_1, λ_2. Therefore the affine transformation has 6 *DOF*. Note that the ratio of λ_1, λ_2 corresponds to a non-isotropic scaling along the axis specified by φ.

Data sets and learning setup Figure 4.7 shows sample image pairs with different transformations. To experimentally investigate how well different types of transformations can be learned by the network, we first note that *CNNs* are inherently able to deal with patch translations due to the convolutional nature. Therefore we identify the following

Data set	Rotation θ [°]
R0-10	$\|\theta\| \in [0, 10]$
R10-20	$\|\theta\| \in [10, 20]$
R20-30	$\|\theta\| \in [20, 30]$
R30-40	$\|\theta\| \in [30, 40]$
R40-50	$\|\theta\| \in [40, 50]$
R50-60	$\|\theta\| \in [50, 60]$

a Isometry data sets.

Data set	Scale
S0.0-0.1	$s \in 1 \pm 0.1$
S0.1-0.2	$s \in [1.1, 1.2] \cup \frac{1}{[1.2, 1.1]}$
S0.2-0.3	$s \in [1.2, 1.3] \cup \frac{1}{[1.3, 1.2]}$
S0.3-0.4	$s \in [1.3, 1.4] \cup \frac{1}{[1.4, 1.3]}$
S0.4-0.5	$s \in [1.4, 1.5] \cup \frac{1}{[1.5, 1.4]}$

b Similarity data sets. Rotation is fixed at $|\theta| \in [0, 20]$.

Data set	Shearing angle φ [°]	Scale Ratio $\mathbf{r} = \lambda_1/\lambda_2$
A1	$\|\varphi\| \in [0, 10]$	$r \in [1.1, 1.4] \cup \frac{1}{[1.4, 1.1]}$
A2	$\|\varphi\| \in [10, 20]$	$r \in [1.3, 1.8] \cup \frac{1}{[1.8, 1.3]}$
A3	$\|\varphi\| \in [15, 35]$	$r \in [1.8, 3.0] \cup \frac{1}{[3.0, 1.8]}$

c Affine data sets. Rotation and global scale are fixed at $|\theta| \in [0, 30]$, $s \in [\frac{1}{1.2}, 1.2]$.

Table 4.1: Data sets for learning patch transformations. Each data set contains 200 image pairs, we generate separate training and test sets

Network	Data sets used
CNN-R10	R0-10
CNN-R20	R0-10, R10-20
CNN-R30	R0-10, R10-20 R20-30
CNN-R40	R0-10, R10-20 R20-30, R30-40
CNN-R50	R0-10, R10-20 R20-30, R30-40 R40-50
CNN-R60	R0-10, R10-20 R20-30, R30-40 R40-50, R50-60

a Isometry networks.

Network	Data sets used
CNN-S0.1	S0.0-0.1
CNN-S0.2	S0.0-0.1, S0.1-0.2
CNN-S0.3	S0.0-0.1, S0.1-0.2 S0.2-0.3
CNN-S0.4	S0.0-0.1, S0.1-0.2 S0.2-0.3, S0.3-0.4
CNN-S0.5	S0.0-0.1, S0.1-0.2 S0.2-0.3, S0.3-0.4 S0.4-0.5

b Similarity networks.

Network	Data sets used
CNN-A1	A1
CNN-A12	A1, A2
CNN-A123	A1, A2, A3

c Affine networks.

Table 4.2: Networks for learning patch transformations. The networks for each subtype of transformation are trained on gradually increasing data size.

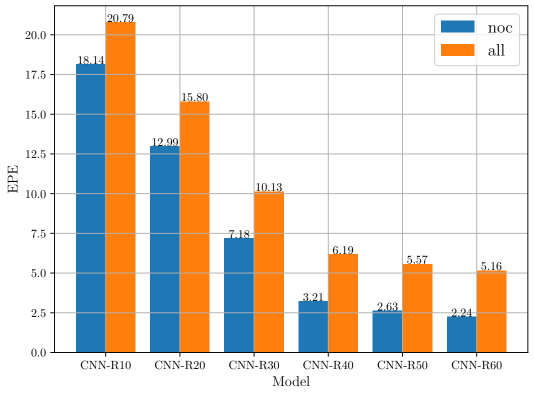
interesting subtypes of transformations: rotation, isotropic scaling and non-isotropic scaling. These correspond to isometries, similarities and affine transformations respectively, modulo translation.[3] For each subtype, we generate separate training and test sets with

[3]For good measure and to make training data more interesting, we include moderate translations of up to ±10 pixels in x and y direction by setting $|t_x|, |t_y| \in [0, 10]$ for all data sets.

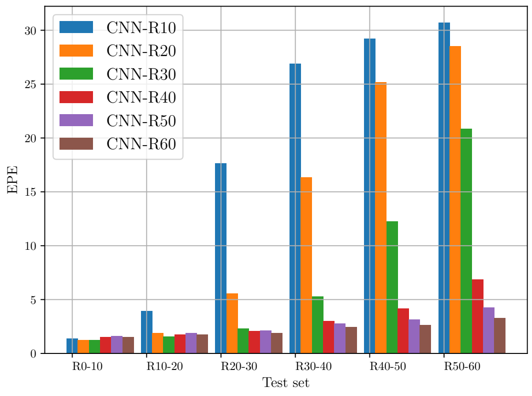
gradually increasing transformations, see table 4.1. For example, data set R40-50 contains rotations between 40 and 50 degrees in positive and negative direction, data set S0.3-0.4 contains scale changes between 1.3 and 1.4 (scaling up) and $\frac{1}{1.4} = 0.71$ and $\frac{1}{1.3} = 0.76$ (scaling down). Each data set consists of 200 image pairs.

The networks are trained by gradually increasing the data size, see table 4.2. For instance, the network CNN-R10 sees rotations of up to $\pm 10°$ during training, whereas the network CNN-R40 sees rotations of up to $\pm 40°$. Networks using only one data set, i.e. CNN-R10, CNN-S0.1 and CNN-A1, train on all 200 image pairs. To keep training times reasonable, we randomly select a subset of 50% of each data set for all other networks. For example, the network CNN-R40 uses 4 data sets, and we randomly select 100 pairs from each set. In total, CNN-R40 trains on 400 image pairs.

Evaluation We are interested in the matching performance of the descriptors only, hence we do not use *CRF* inference but rather compute the *WTA* solution. We present the overall End Point Error (EPE) on all as well as on non-occluded pixels, where we evaluate each network model on all of its test sets combined. E.g. each isometry network is evaluated on all 6 isometry test sets. Note that during training occluded regions are masked, therefore *EPE* (noc) is more indicative of the quality of the descriptors than *EPE* all. Moreover, we also present a breakdown of the network performance per test set in order to investigate at which point the performance breaks down.



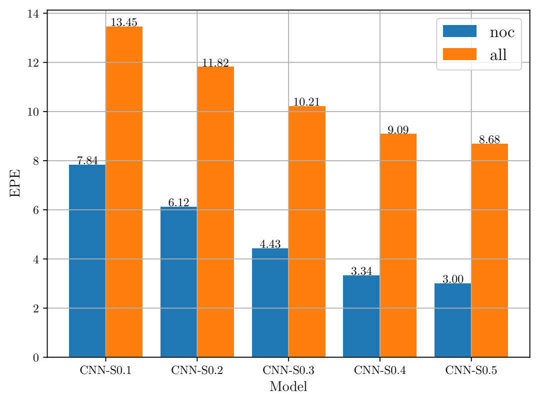
(a) Overall *EPE* for different networks on all isometry test sets combined. The more data the network has seen during training, the better its performance.



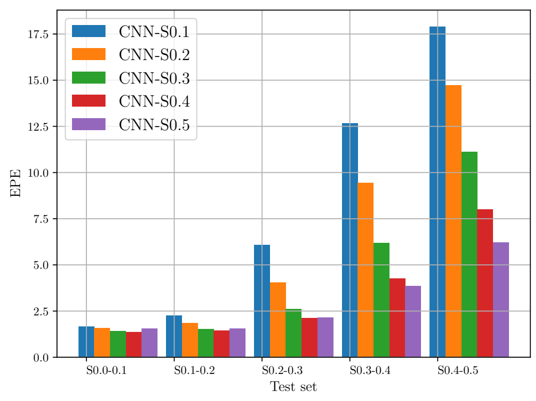
(b) *EPE* (noc) per network and test set. The network trained on rotations up to $\pm 60°$ yields consistent good results.

Figure 4.8: Results for learning isometries.

Figure 4.8 depicts the results of learning rotations. It shows that the networks which saw more rotations during training in general perform better. The detailed breakdown in fig. 4.8(b) shows that our networks can adapt to rotations up to $\pm 60°$, provided the training data contains such transformations. In particular, the network CNN-R60 yields

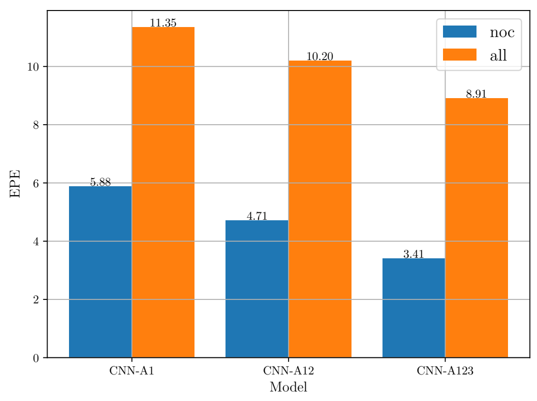


(a) Overall *EPE* for different networks on all similarity test sets.

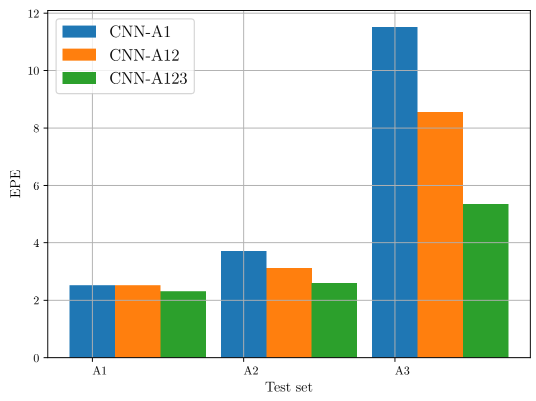


(b) *EPE* (noc) per network and test set. Scale changes of more than 1.35 and less than $\frac{1}{1.35}$ are challenging.

Figure 4.9: Results for learning similarities.



(a) Overall *EPE* for different networks on all affine test sets.



(b) *EPE* (noc) per network and test set.

Figure 4.10: Results for learning affine transformations.

consistent good results. It is interesting to note that a network trained on rotations up to θ° performs reasonably on test data with rotations up to $\theta+10^\circ$ before breaking down. The results show that the limiting factor for learning rotations is the amount of transformation in the training data. Even inter-frame rotations of more than 45°, which are unlikely to occur in real-world data, can be matched with good quality.

The results for learning scale changes are depicted in fig. 4.9. Whereas the overall *EPE* decreases with the amount of training data, we see in the detailed analysis fig. 4.9(b) that learning scale changes is more difficult than learning rotations. In particular, at larger scale changes the error grows more rapidly. This indicates that the current network architecture need to be improved to effectively deal with scale changes.

The same holds true for learning affine transformations, see fig. 4.10. This is intuitive, since the affine transformation differs from the similarity only in the fact that the scaling is non-isotropic. From these results, the most obvious way to learn better descriptors is to adapt the network architecture such that it can better handle scale changes.

4.3.2 Comparison of our Models

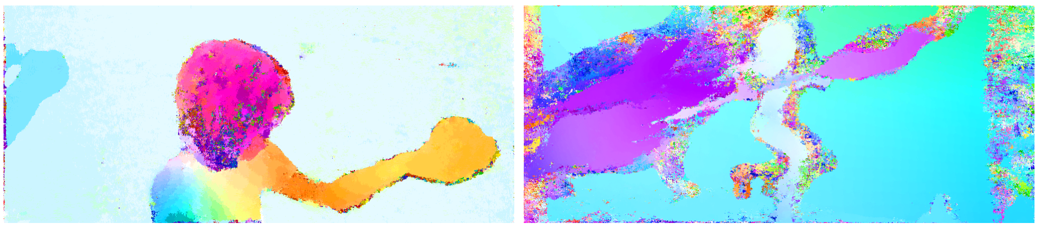
Next, we compare different variants of our model on the Sintel optical flow dataset [Butler et al., 2012]. In total the benchmark consists of 1064 training images and 564 test images. For *CNN* learning we use a subset of 20% of the training images, sampled evenly from all available scenes. For evaluation, we use a subset of 40% of the training images.

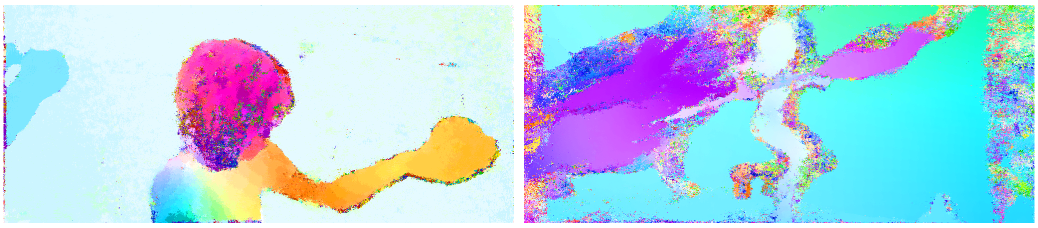
To investigate the performance of our model, we conduct the following experiments: First, we investigate the influence of the size of the *CNN*, and second we investigate the effect of quantizing the learned features. Additionally, we evaluate both the *WTA* solution eq. (4.2), and the *CRF* model eq. (4.3). To assess the effect of quantization, we evaluate the local flow model a) as it was trained, and b) QQ, i.e., with quantized descriptors both in the min-projection step as well as in the outer problem on $c^{\mathrm{U}}, c^{\mathrm{V}}$ respectively. For *CRF* inference the updates eq. (4.10) and eq. (4.11) amount to solving a min-projection step with additional cost offsets. F and Q indicate how this min-projection step is computed. *CRF* parameters are fixed at $\alpha = 8.5$ (eq. (4.3)), $\tau_1 = 0.25$, $\tau_2 = 25, \delta = 8$ (fig. 4.1) for all experiments and we run 8 inner and 5 outer iterations. Table 4.3 summarizes the comparison of different variants of our model. We see that the *WTA* solution of model FQ performs similarly to FF, while being much faster to train and evaluate, cf. table 4.4. In particular, model FQ performs better than QQ, which was trained with the straight

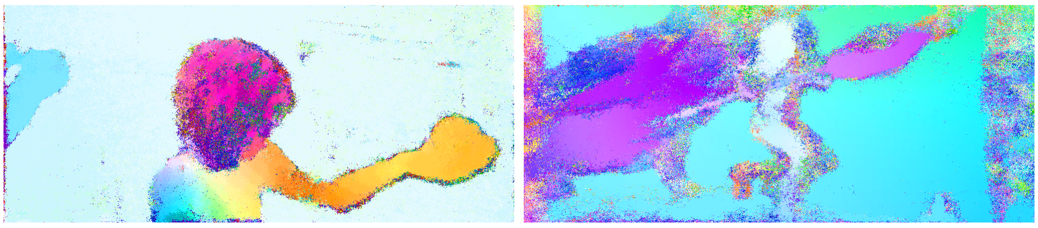
		Local Flow Model (WTA)		**CRF**	
Train	**#Layers**	**as trained** noc (all)	**QQ** noc (all)	**F** noc (all)	**Q** noc (all)
FF	5	5.25 (10.38)	10.45 (15.67)	1.58 (4.48)	1.64 (4.87)
	7	4.72 (10.04)	9.43 (14.93)	1.53 (4.32)	1.61 (4.70)
	9	–[1]	–[1]	–[1]	–[1]
FQ	5	6.15 (11.36)	11.43 (16.78)	–[2]	1.63 (4.62)
	7	5.62 (10.98)	10.15 (15.70)	–[2]	1.65 (4.62)
	9	5.62 (11.13)	9.87 (15.52)	–[2]	1.64 (4.69)
QQ	5	same as QQ	9.63 (14.80)	–[2]	1.72 (4.91)
	7	same as QQ	9.75 (15.23)	–[2]	1.66 (4.78)
	9	same as QQ	9.72 (15.31)	–[2]	1.72 (4.85)

Table 4.3: Comparison of our models on a representative validation set at scale $\frac{1}{2}$. We present the *EPE* for non-occluded (noc) and all pixels on Sintel clean.
[1]Omitted due to very long training time.
[2]Not applicable.

through estimator of the gradient. If we switch to QQ for evaluation, we see a drop in performance for models FF and FQ. This is to be expected, because we now evaluate costs differently than during training. Interestingly, our *CRF* model yields similar performance regardless whether we use F or Q for computing the costs. Figure 4.11 depicts sample


(a) Model WTA-FF, *EPE* for left / right image: 1.8 / 19.7


(b) Model WTA-FQ, *EPE* for left / right image: 2.1 / 20.7


(c) Model WTA-QQ, *EPE* for left / right image: 3.4 / 24.1

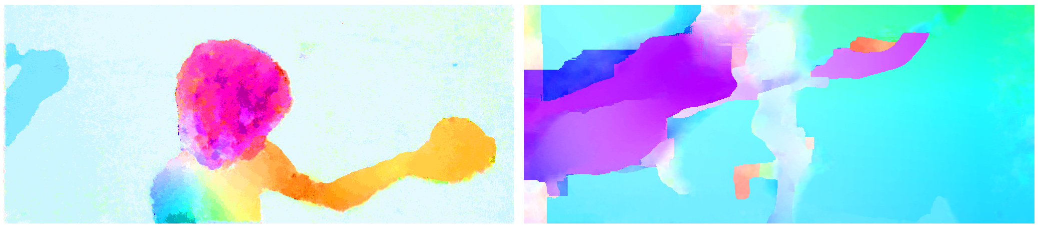
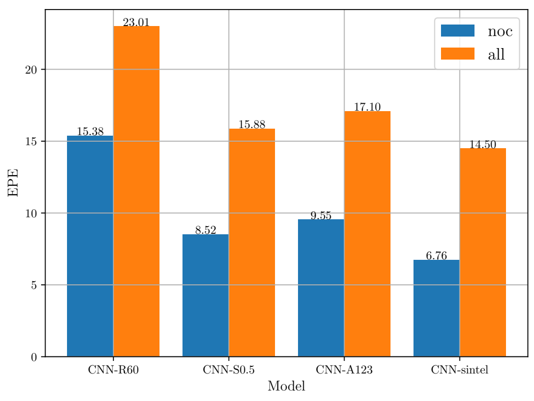

(d) Model CRF-Q, *EPE* for left / right image: 0.7 / 9.3

Figure 4.11: Results of the *WTA* solution for the models FF (a), FQ (b) and QQ (c), as well as the *CRF*-Q solution (d).

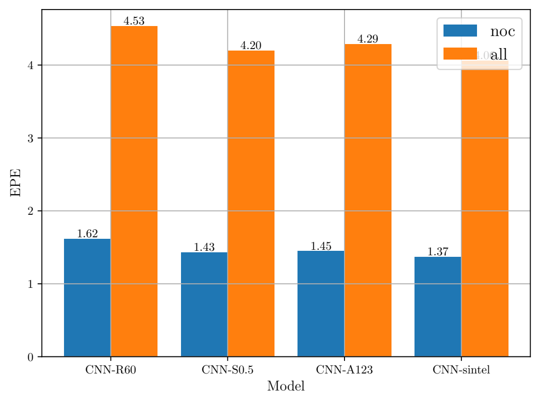
results of the *WTA* solution for the three models FF, FQ and QQ. The images reflect the numbers of table 4.3, FF and FQ perform similar whereas QQ is considerably worse. Note that if matching is possible, i.e. in non-occluded image regions, the simple *WTA* solution performs already quite well, with few spurious wrong matches (see left column in fig. 4.11). Most of these wrong matches are taken care of by the *CRF*. The remaining errors of the *CRF* solution fig. 4.11(d) are mainly in occluded regions, where any two-frame method is forced to hallucinate.

Table 4.3 shows that model FQ delivers a good performance while being much faster to train and evaluate than FF. Therefore we fix the training model to FQ for the subsequent experiments. For evaluation, we fix FQ for computing the *WTA* solution and Q for *CRF* inference.

We also conduct an experiment to see how the networks that were trained on synthetic data (see section 4.3.1) perform on real data. To that end, we evaluate the homography networks from the previous section on the Sintel dataset. We choose the clean version of the benchmark, since our synthetic training data does not contain motion blur, lens distortion, fog and other effects present in the final version of Sintel. A network trained on Sintel clean is used for comparison. It is no surprise that simple *WTA* matching works



(a) Overall *EPE* of *WTA* solution on Sintel clean.



(b) Overall *EPE* of *CRF*-Q solution on Sintel clean.

Figure 4.12: Comparison of the affine transformation networks on the Sintel dataset. For comparison, the model CNN-sintel was trained specifically on the Sintel clean data.

best when the training data is similar to the test data. We think that the performance of the CNN-scale0.5 network is remarkable, given the fact that it was trained only on simple scaled and rotated random patches. When we add the *CRF*, the performance differences get even smaller. The experiments indicate that it is feasible to learn general descriptors for the purpose of matching from completely synthetic data.

4.3.3 Runtime

The main reason for quantizing the descriptors is speed. For *CRF* inference, we need to compute the min-projection on the 4D cost function twice per outer iteration, see Alg. 4.1. We show an exact breakdown of the timings for $D = 128$ on high resolution images in table 4.4, computed on a Intel i7 6700K and a Nvidia Titan X.

The column *WTA* refers to computing the solution of the local model on the cost volumes $c^{\mathrm{U}}, c^{\mathrm{V}}$, see eq. (4.6). Full model is the *CRF* inference, see section 4.2.2. We see that we can reach a significant speed-up by using binary descriptors and Hamming

Method	**Feature Extraction**	**WTA**	**Full Model**
FF	0.04 – 0.08	4.25	24.8
FQ	0.04 – 0.08	1.82	-
QQ	0.04 – 0.08	0.07	3.2
[Xu et al., 2017] ($\frac{1}{3}$ res.)	0.02	0.06	3.4
QQ ($\frac{1}{3}$ res.)	0.004 – 0.008	0.007	0.32

Table 4.4: Timings of the building blocks (seconds).

distance for computing intensive calculations. For comparison, we also report the runtime of [Xu et al., 2017], who, at the time of writing, report the fastest execution time on Sintel. We point out that our *CRF* inference on full resolution images takes about the same time as their method, which constructs and optimizes the cost function at $\frac{1}{3}$ resolution.

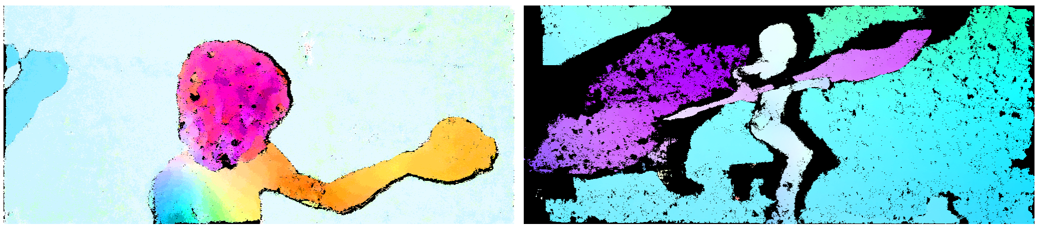
4.3.4 Benchmark Results

We compare our method on the Sintel final benchmark. Our *CRF* is effective in inferring the match if matching is possible, but it does not perform well in occluded regions. As it is common in many other optical flow methods, we apply a postprocessing aiming specifically at inpainting occluded areas. First, we compute the forward flow between I^1 and I^2 and the backward flow between I^2 and I^1. Using a forward-backward check, we mask a pixel as unmatched/occluded if the forward and backward flow do not agree. Then we feed all surviving matches to EpicFlow [Revaud et al., 2015], a method specialized in inpainting flow fields. A sample input to Epicflow is depicted in fig. 4.13(a). Pixels where forward and backward flow do not agree are masked in black. In case of large occlusions, e.g. right image of fig. 4.13(a), we see that the forward-backward check is effective in detecting occluded areas. The result of the EpicFlow inpainting is shown in fig. 4.13(b). Compared to the *EPE* numbers of pure *CRF* inference (see fig. 4.11), we can decrease the *EPE* substantially.

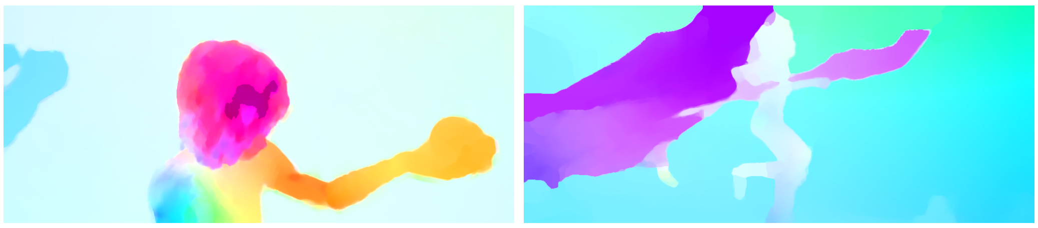
The results on the Sintel final test set are shown in table 4.5. Whereas we are competitive on the *EPE* all metric, our method outperforms many well known algorithms on the *EPE* noc metric. We think that this shows that our network learns robust features that are well-suited for image matching. The discrepancy between the all and noc metric suggests that we can improve our results by using a more sophisticated handling of occluded image regions.

4.4 Conclusion

We showed that both learning and *CRF* inference of the optical flow cost function on high resolution images is tractable. We circumvent the excessive memory requirements of the full 4D cost volume by a min-projection. This reduces the space complexity from quadratic to linear in the search range. To efficiently compute the cost function, we learn



(a) Initialization for EpicFlow. Unmatched/occluded pixels are masked in black.



(b) Result of EpicFlow inpainting, *EPE* for left / right image: 0.2 / 4.0

Figure 4.13: Postprocessing with EpicFlow results in substantial decrease of the *EPE*. For reference, *EPE* with pure *CRF* inference is 0.7 / 9.3 respectively, see fig. 4.11. The forward-backward check masks mainly the occluded regions, see right image in (a), which afterwards get inpainted by EpicFlow.

Method	**EPE all** [**Rank**]	**EPE noc** [**Rank**]
DCFlow [Xu et al., 2017]	5.119 [2]	2.283 [3]
FlowFields+ [Bailer et al., 2017]	5.707 [13]	2.684 [15]
FullFlow [Chen and Koltun, 2016]	5.895 [21]	2.838 [22]
FlowNet2 [Ilg et al., 2017]	6.016 [23]	2.977 [28]
DiscreteFlow [Menze et al., 2015]	6.077 [29]	2.937 [24]
Ours CRF-Q + EpicFlow	6.053 [26]	2.618 [10]

Table 4.5: Results on the Sintel final dataset. Our method performs particularly well on the non-occluded metric, indicating that the learned descriptors are of high quality.

binary descriptors with a new hybrid learning scheme, that outperforms the previous state-of-the-art straight-through estimator of the gradient.

5 Summary

In this work, we have considered the dense image matching problem. We presented a general energy minimization formulation consisting of a data term and a regularization term in order to solve the ill-posed problem. Motivated by the theories of differential geometry, we introduced a regularization term based on the inner geometry of the solution manifold. We derived the surface area functional under orthographic and perspective projection, which enables to penalize surface area in the parametric domain, i.e. on the image plane. Being able to formulate the problem on the image plane bypasses the need for an explicit representation of surface geometry and is the key for highly efficient parallel algorithms. We showed that the non-convex area functional becomes convex under a suitable reparameterization, and we formulated a dense stereo problem using surface area as regularizer. We compared our area regularization to standard Total Variation (TV) regularization, and we showed theoretically and experimentally that our stereo model is not affected by the staircaising effect of TV. The experiments confirm quantitatively that area regularization leads to higher quality depthmaps and better 3D-reconstructions.

In the context of intensity image reconstruction for event cameras, we used the manifold of active events to steer regularization by formulating the entire variational problem on the manifold. Using a variant of the ROF-model for image denoising, we showed that computing the TV-norm under the metric of the manifold introduces anisotropy based on the geometry of the manifold. Applied to the problem of intensity image reconstruction for event cameras, this approach results in sharper edges, higher contrast and more graceful grayvalue variations in the reconstructed images. Because the problem is still formulated on the image plane, our algorithm can be trivially accelerated by massively parallel Graphics Processing Units (GPUs) and we reach a time resultion of up to 500 reconstructed frames per second.

Concerning the low-level problem of dense optical flow estimation, we showed that Convolutional Neural Network (CNN)-based learning of descriptors for the four-dimensional optical flow cost function is tractable. To that end, we proposed a min-projection, which reduces the memory complexity from quadratic to linear in the size of the search range. We tackled the high computational demands by introducing a binarization step in the learning, and we proposed a hybrid learning scheme which yields better results than previous state of the art in learning binary *CNNs*. Using a Conditional Random Field (CRF) for robust inference, we adapted the idea of dimensionality reduction by decomposing the graphical model into smaller subproblems, where the horizontal and vertical subproblems correspond to stereo problems and the inter-plane subproblems correspond to a min-projection with additional cost offsets. Our approach allows to efficiently solve the *CRF* with the full quadratic label space at linear memory complexity.

5.1 Outlook

In contrast to *TV* regularization, the minimal area regularizer has no bias towards fronto-parallel surfaces. However, penalizing area is distance dependent and has a natural bias towards shrinking the surface. An alternative would be to penalize surface curvature, but it turns out that the pullback of curvature under perspective projection is a rather complicated expression. Moreover, while the surface area is a first-order functional which can be reparameterized, i.e. convexified, rather easily, finding a suitable convexification of the higher-order nonlinear curvature functional turns out to be much harder. Whereas it is possible to optimize a non-convex and non-linear functional using a variant of the primal-dual algorithm [Valkonen, 2014], we did not succeed in implementing a stereo model with curvature regularization. In general, curvature regularization is still an open problem [Yashtini and Kang, 2015, Zhang and Chen, 2016] due to the difficulty of the higher-order non-linear functional.

Recent results in learning descriptors for stereo and optical flow has shown that problem-specific features learned by *CNNs* are clearly superior to the hand-crafted features that have been used in the past. However, our experiments on synthetic training data indicate that there is potential for improvement. We showed that while a *CNN* can learn rotations very well, standard network architectures have difficulties in learning scale changes. [Zagoruyko and Komodakis, 2015] report that a multiscale-strategy, where the network explicitly learns simultaneously on small and larger image patches, yields good results. We conducted experiments with undersampling and varying the size of the receptive field using dilated convolutions. We found that whereas larger receptive fields result in more robust matching in difficult areas, i.e. distortions, motion blur, little texture etc., they come at a loss of matching accuracy at fine details. This is in a sense similar to the problem of the coarse-to-fine warping scheme used in continuous methods. We think that networks which explicitly take into account different scales are a promising future research direction.

Another fundamental problem in dense image matching are occlusions. Explicitly modeling occlusions in dense correspondence problems turns out to be difficult and computationally very demanding [Unger et al., 2012, Ayvaci et al., 2012]. It seems that incorporating occlusions is most helpful as part of a general scene understanding system [Hur and Roth, 2016]. Eventually the findings from such higher-level methods can be used as a guideline on how to deal with occlusion in low-level correspondence problems.

A List of Acronyms

AD	Absolute Differences
CMOS	Complementary Metal-Oxide-Semiconductor
CNN	Convolutional Neural Network
CRF	Conditional Random Field
DMM	Dual Minorize-Maximize
DOF	Degrees of Freedom
DSLR	Digital Single-Lens Reflex
DVS	Dynamic Vision Sensor
EPE	End Point Error
FAST	Features from Accelerated Segment Test
FISTA	Fast Iterative Shrinkage Thresholding Algorithm
GPU	Graphics Processing Unit
LP	Linear Programming
MAP	Maximum a Posteriori
MRF	Markov Random Field
NCC	Normalized Cross-Correlation
PPA	Proximal Point Algorithm
RANSAC	Random Sample Consensus
RMS	Root Mean Square
SAD	Sum of Absolute Differences
SGM	Semiglobal Matching
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
TGV	Total Generalized Variation
TRW	Tree-Reweighted Max-Product Message Passing

TRW-S	Sequential Tree-Reweighted Message Passing
TV	Total Variation
UAV	Unmanned Aerial Vehicle
WTA	Winner-Takes-All

Bibliography

[Arora et al., 2012] Arora, C., Banerjee, S., Kalra, P., and Maheshwari, S. (2012). Generic cuts: An efficient algorithm for optimal inference in higher order mrf-map. In *Proc. European Conference on Computer Vision (ECCV)*. (page 7)

[Ayvaci et al., 2012] Ayvaci, A., Raptis, M., and Soatto, S. (2012). Sparse occlusion detection with optical flow. *International Journal of Computer Vision (IJCV)*, 97(3):322–338. (page 128)

[Bailer et al., 2012] Bailer, C., Finckh, M., and Lensch, H. P. A. (2012). Scale Robust Multi View Stereo. In *Proc. European Conference on Computer Vision (ECCV)*. (page 77)

[Bailer et al., 2015] Bailer, C., Taetz, B., and Stricker, D. (2015). Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. In *Proc. International Conference on Computer Vision (ICCV)*. (page 108)

[Bailer et al., 2017] Bailer, C., Taetz, B., and Stricker, D. (2017). Optical flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation. *CoRR*, abs/1703.02563. (page 126)

[Balzer and Soatto, 2014] Balzer, J. and Soatto, S. (2014). Second-order Shape Optimization for Geometric Inverse Problems in Vision. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 75)

[Bardow et al., 2016] Bardow, P., Davison, A., and Leutenegger, S. (2016). Simultaneous optical flow and intensity estimation from an event camera. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 95, 96, 98, 102, 103, 104)

[Barua et al., 2016] Barua, S., Miyatani, Y., and Veeraraghavan, A. (2016). Direct face detection and video reconstruction from event cameras. In *Winter Conference on Applications of Computer Vision (WACV)*. (page 96)

[Beck and Teboulle, 2009] Beck, A. and Teboulle, M. (2009). A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on imaging sciences*, 2(1):183–202. (page 9, 25)

[Bengio et al., 2013] Bengio, Y., Léonard, N., and Courville, A. C. (2013). Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432. (page 108, 109, 114)

[Benosman et al., 2014] Benosman, R., Clercq, C., Lagorce, X., Ieng, S. H., and Bartolozzi, C. (2014). Event-based visual flow. *Transactions on Neural Networks and Learning Systems*, 25(2):407–417. (page 95, 96, 98)

[Bertsekas, 1999] Bertsekas, D. (1999). *Nonlinear programming*. Athena Scientific optimization and computation series. Athena Scientific. (page 8, 33)

[Besse et al., 2012] Besse, F., Rother, C., Fitzgibbon, A., and Kautz, J. (2012). PMBP: PatchMatch Belief Propagation for Correspondence Field Estimation. In *Proc. British Machine Vision Conference (BMVC)*. (page 77, 78)

[Bleyer et al., 2011] Bleyer, M., Rhemann, C., and Rother, C. (2011). PatchMatch Stereo - Stereo Matching with Slanted Support Windows. In *Proc. British Machine Vision Conference (BMVC)*. (page 77, 78)

[Bodis-Szomoru et al., 2014] Bodis-Szomoru, A., Riemenschneider, H., and Gool, L. V. (2014). Fast, Approximate Piecewise-Planar Modeling Based on Sparse Structure-from-Motion and Superpixels. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 77)

[Boros and Hammer, 2002] Boros, E. and Hammer, P. L. (2002). Pseudo-Boolean optimization. *Discrete Applied Mathematics*, 1-3(123):155–225. (page 28, 29)

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press. (page 8, 13)

[Boykov and Kolmogorov, 2004] Boykov, Y. and Kolmogorov, V. (2004). An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124–1137. (page 6, 29)

[Boykov et al., 1998] Boykov, Y., Veksler, O., and Zabih, R. (1998). Markov random fields with efficient approximations. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 6, 29)

[Boykov et al., 2001] Boykov, Y., Veksler, O., and Zabih, R. (2001). Fast approximate energy minimization via graph cuts. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 23(11):1222–1239. (page 6, 29)

[Bradley et al., 2008] Bradley, D., Boubekeur, T., and Heidrich, W. (2008). Accurate multi-view reconstruction using robust binocular stereo and surface meshing. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 77)

[Bredies et al., 2010] Bredies, K., Kunisch, K., and Pock, T. (2010). Total generalized variation. *SIAM Journal on Imaging Sciences*, 3(3):492–526. (page 8)

[Brox et al., 2004] Brox, T., Bruhn, A., Papenberg, N., and Weickert, J. (2004). High accuracy optical flow estimation based on a theory for warping. In *Proc. European Conference on Computer Vision (ECCV)*. (page 8)

[Butler et al., 2012] Butler, D. J., Wulff, J., Stanley, G. B., and Black, M. J. (2012). A naturalistic open source movie for optical flow evaluation. In *Proc. European Conference on Computer Vision (ECCV)*. (page 122)

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. In *Proc. European Conference on Computer Vision (ECCV)*. (page 108)

[Campbell et al., 2008] Campbell, N. D. F., Vogiatzis, G., Hern, C., and Cipolla, R. (2008). Using Multiple Hypotheses to Improve Depth-Maps for Multi-View Stereo. In *Proc. European Conference on Computer Vision (ECCV)*. (page 76)

[Chambolle, 2004] Chambolle, A. (2004). An algorithm for total variation minimization and applications. *Journal of Mathematical Imaging and Vision*, 20(1-2):89–97. (page 88, 101)

[Chambolle and Pock, 2011] Chambolle, A. and Pock, T. (2011). A first-order primal-dual algorithm for convex problems with applications to imaging. *Journal of Mathematical Imaging and Vision*, 40(1):120–145. (page 9, 26, 89, 101)

[Chambolle and Pock, 2016] Chambolle, A. and Pock, T. (2016). An introduction to continuous optimization for imaging. *Acta Numerica*, 25:161–319. (page 8)

[Chen and Koltun, 2016] Chen, Q. and Koltun, V. (2016). Full flow: Optical flow estimation by global optimization over regular grids. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 108, 115, 126)

[Chen et al., 2015] Chen, Z., Sun, X., Wang, L., Yu, Y., and Huang, C. (2015). A deep visual correspondence embedding model for stereo matching costs. In *Proc. International Conference on Computer Vision (ICCV)*. (page 113)

[Cheng et al., 2000] Cheng, L.-T., Burchard, P., Merriman, B., and Osher, S. (2000). Motion of curves constrained on surfaces using a level set approach. *Journal of Computational Physics*, 175(2):604–644. (page 98)

[Choi and Rutenbar, 2012] Choi, J. and Rutenbar, R. A. (2012). Hardware implementation of mrf map inference on an fpga platform. In *Field Programmable Logic and Applications (FPL)*, pages 209–216. (page 7)

[Civera et al., 2008] Civera, J., Davison, A. J., and Montiel, J. (2008). Inverse depth parametrization for monocular SLAM. *Transactions on Robotics*, 24(5):932–945. (page 78)

[Cook et al., 2011] Cook, M., Gugelmann, L., Jug, F., Krautz, C., and Steger, A. (2011). Interacting maps for fast visual interpretation. In *International Joint Conference on Neural Networks (IJCNN)*. (page 96)

[Cook et al., 1998] Cook, W. J., Cunningham, W. H., Pulleyblank, W. R., and Schrijver, A. (1998). *Combinatorial Optimization*. John Wiley & Sons, Inc., New York, USA. (page 29)

[Courbariaux and Bengio, 2016] Courbariaux, M. and Bengio, Y. (2016). Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1. *CoRR*, abs/1602.02830. (page 108, 109)

[Delaunoy and Pollefeys, 2014] Delaunoy, A. and Pollefeys, M. (2014). Photometric Bundle Adjustment for Dense Multi-View 3D Modeling. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 75)

[Delbruck and Lichtsteiner, 2007] Delbruck, T. and Lichtsteiner, P. (2007). Fast sensory motor control based on event-based hybrid neuromorphic-procedural system. In *International Symposium on Circuits and Systems*. (page 94)

[Delong and Boykov, 2008] Delong, A. and Boykov, Y. (2008). A scalable graph-cut algorithm for nd grids. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 6)

[do Carmo, 1992] do Carmo, M. P. (1992). *Riemannian Geometry*. Mathematics (Boston, Mass.). Birkhäuser, 2nd edition. (page 35)

[Elias et al., 1956] Elias, P., Feinstein, A., and Shannon, C. (1956). A note on the maximum flow through a network. *Transactions on Information Theory*, 2(4):117–119. (page 6, 29)

[Fan and Ferrie, 2010] Fan, S. and Ferrie, F. P. (2010). Photo Hull regularized stereo. *Image and Vision Computing*, 28(4):724–730. (page 76)

[Felzenszwalb and Huttenlocher, 2006] Felzenszwalb, P. F. and Huttenlocher, D. P. (2006). Efficient belief propagation for early vision. *International Journal of Computer Vision (IJCV)*, 70(1):41–54. (page 108, 115)

[Fix et al., 2014] Fix, A., Wang, C., and Zabih, R. (2014). A primal-dual algorithm for higher-order multilabel markov random fields. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 7)

[Ford and Fulkerson, 1956] Ford, L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404. (page 6, 29)

[Furukawa and Ponce, 2010] Furukawa, Y. and Ponce, J. (2010). Accurate, dense, and robust multiview stereopsis. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1362–1376. (page 77)

[Gadot and Wolf, 2016] Gadot, D. and Wolf, L. (2016). Patchbatch: A batch augmented loss for optical flow. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 108, 111)

[Gallego et al., 2015] Gallego, G., Forster, C., Mueggler, E., and Scaramuzza, D. (2015). Event-based camera pose tracking using a generative event model. *CoRR*, abs/1510.01972. (page 95, 96)

[Gallup, 2014] Gallup, D. (2014). 3D Reconstruction from Accidental Motion. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 76)

[Goesele et al., 2006] Goesele, M., Curless, B., and Seitz, S. (2006). Multi-View Stereo Revisited. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 76)

[Graber et al., 2011] Graber, G., Pock, T., and Bischof, H. (2011). Online 3D reconstruction using convex optimization. In *International Conference on Computer Vision (ICCV) Workshops*. (page 77, 84)

[Greig et al., 1989] Greig, D. M., Porteous, B. T., and Seheult, A. H. (1989). Exact maximum a posteriori estimation for binary images. *Journal of the Royal Statistical Society. Series B (Methodological)*, 51(2):271–279. (page 6)

[Grinfeld, 2013] Grinfeld, P. (2013). *Introduction to Tensor Analysis and the Calculus of Moving Surfaces*. Springer New York. (page 35)

[Güney and Geiger, 2016] Güney, F. and Geiger, A. (2016). Deep discrete flow. In *Proc. Asian Conference on Computer Vision (ACCV)*. (page 108, 111)

[Hartley and Zisserman, 2004] Hartley, R. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition. (page 78)

[Hartmann et al., 2013] Hartmann, J., Klüssendorff, J. H., and Maehle, E. (2013). A comparison of feature descriptors for visual slam. In *European Conference on Mobile Robots*. (page 96)

[Heber and Pock, 2014] Heber, S. and Pock, T. (2014). Shape from Light Field meets Robust PCA. In *Proc. European Conference on Computer Vision (ECCV)*. (page 78)

[Heise et al., 2013] Heise, P., Klose, S., Jensen, B., and Knoll, A. (2013). PM-Huber: PatchMatch with Huber Regularization for Stereo Matching. In *Proc. International Conference on Computer Vision (ICCV)*. (page 78)

[Hernández and Schmitt, 2004] Hernández, C. and Schmitt, F. (2004). Silhouette and stereo fusion for 3D object modeling. *Computer Vision and Image Understanding*, 96(3):367–392. (page 75)

[Hirschmüller, 2005] Hirschmüller, H. (2005). Accurate and efficient stereo processing by semi-global matching and mutual information. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 110)

[Hochbaum, 2001] Hochbaum, D. S. (2001). An efficient algorithm for image segmentation, markov random fields and related problems. *J. ACM*, 48(4):686–701. (page 29)

[Hu and Mordohai, 2012] Hu, X. and Mordohai, P. (2012). Least Commitment, Viewpoint-Based, Multi-view Stereo. In *Proc. International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*. (page 76)

[Hur and Roth, 2016] Hur, J. and Roth, S. (2016). Joint optical flow and temporally consistent semantic segmentation. In *European Conference on Computer Vision (ECCV) Workshops*. (page 129)

[Hurkat et al., 2015] Hurkat, S., Choi, J., Nurvitadhi, E., Martínez, J. F., and Rutenbar, R. A. (2015). Fast hierarchical implementation of sequential tree-reweighted belief propagation for probabilistic inference. In *Field Programmable Logic and Applications (FPL)*, pages 1–8. (page 7)

[Ilg et al., 2017] Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., and Brox, T. (2017). Flownet 2.0: Evolution of optical flow estimation with deep networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 126)

[Ishikawa, 2003] Ishikawa, H. (2003). Exact optimization for markov random fields with convex priors. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 25(10):1333–1336. (page 29)

[Ishikawa and Geiger, 1998] Ishikawa, H. and Geiger, D. (1998). Segmentation by grouping junctions. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 6)

[Kappes et al., 2015] Kappes, J. H., Andres, B., Hamprecht, F. A., Schnörr, C., Nowozin, S., Batra, D., Kim, S., Kausler, B. X., Kröger, T., Lellmann, J., Komodakis, N., Savchynskyy, B., and Rother, C. (2015). A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision (IJCV)*, 115(2):155–184. (page 7)

[Kappes et al., 2012] Kappes, J. H., Savchynskyy, B., and Schnörr, C. (2012). A bundle approach to efficient map-inference by lagrangian relaxation. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 34)

[Kim et al., 2014] Kim, H., Handa, A., Benosman, R., Ieng, S.-H., and Davison, A. (2014). Simultaneous mosaicing and tracking with an event camera. In *Proc. British Machine Vision Conference (BMVC)*. (page 95, 96)

[Klaus et al., 2006] Klaus, A., Sormann, M., and Karner, K. (2006). Segment-Based Stereo Matching Using Belief Propagation and a Self-Adapting Dissimilarity Measure. In *Proc. International Conference on Pattern Recognition (ICPR)*. (page 77)

[Knöbelreiter et al., 2017] Knöbelreiter, P., Reinbacher, C., Shekhovtsov, A., and Pock, T. (2017). End-to-end training of hybrid CNN-CRF models for stereo. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 109, 113)

[Kolmogorov, 2005] Kolmogorov, V. (2005). Primal-dual algorithm for convex markov random fields. Technical Report MSR-TR-2005-117, Microsoft Research. (page 29)

[Kolmogorov, 2006] Kolmogorov, V. (2006). Convergent tree-reweighted message passing for energy minimization. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(10):1568–1583. (page 6, 34, 108, 116)

[Komodakis et al., 2011] Komodakis, N., Paragios, N., and Tziritas, G. (2011). MRF energy minimization and beyond via dual decomposition. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(3):531–552. (page 34)

[Krueger et al., 2008] Krueger, M., Delmas, P., and Gimel'farb, G. L. (2008). Active contour based segmentation of 3d surfaces. In *Proc. European Conference on Computer Vision (ECCV)*. (page 98)

[Lai and Chan, 2011] Lai, R. and Chan, T. F. (2011). A framework for intrinsic image processing on surfaces. *Computer Vision and Image Understanding*, 115(12):1647 – 1661. Special issue on Optimization for Vision, Graphics and Medical Imaging: Theory and Applications. (page 98)

[Lauritzen, 1998] Lauritzen, S. L. (1998). *Graphical Models*. Number 17 in Oxford Statistical Science Series. Oxford Science Publications. (page 27)

[Le et al., 2007] Le, T., Chartrand, R., and Asaki, T. J. (2007). A variational approach to reconstructing images corrupted by poisson noise. *Journal of Mathematical Imaging and Vision*, 27:257–263. (page 101)

[Lee, 2012] Lee, J. M. (2012). *Introduction to Smooth Manifolds*. Graduate Texts in Mathematics. Springer New York, 2nd edition. (page 35, 45, 54)

[Lempitsky et al., 2010] Lempitsky, V., Rother, C., Roth, S., and Blake, A. (2010). Fusion moves for markov random field optimization. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 32(8):1392–1405. (page 6)

[Lewis, 1995] Lewis, J. P. (1995). Fast normalized cross-correlation. *Vision Interface*, 10(1):120–123. (page 3)

[Li and Zucker, 2006a] Li, G. and Zucker, S. W. (2006a). Differential Geometric Consistency Extends Stereo to Curved Surfaces. In *Proc. European Conference on Computer Vision (ECCV)*. (page 78)

[Li and Zucker, 2006b] Li, G. and Zucker, S. W. (2006b). Surface Geometric Constraints for Stereo in Belief Propagation. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 78)

[Lichtsteiner et al., 2008] Lichtsteiner, P., Posch, C., and Delbruck, T. (2008). A 128×128 120 db 15 μs latency asynchronous temporal contrast vision sensor. *Journal of Solid-State Circuits*, 43(2):566–576. (page 94)

[Liu et al., 2009] Liu, Y., Cao, X., Dai, Q., and Xu, W. (2009). Continuous depth estimation for multi-view stereo. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 77, 84)

[Lovász, 1983] Lovász, L. (1983). Submodular functions and convexity. In Bachem, A., Korte, B., and Grötschel, M., editors, *Mathematical Programming The State of the Art: Bonn 1982*, pages 235–257. Springer Berlin Heidelberg. (page 28)

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision (IJCV)*, 60(2):91–110. (page 3)

[Lui et al., 2008] Lui, L. M., Gu, X., Chan, T. F., and Yau, S.-T. (2008). Variational method on riemann surfaces using conformal parameterization and its applications to image processing. *Methods and Applications of Analysis*, 15(4):513–538. (page 98)

[Luo et al., 2016] Luo, W., Schwing, A., and Urtasun, R. (2016). Efficient deep learning for stereo matching. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 10, 108, 111, 113)

[Martinet, 1970] Martinet, B. (1970). Brève communication. régularisation d'inéquations variationnelles par approximations successives. *Revue française d'informatique et de recherche opérationnelle. Série rouge*, 4(R3):154–158. (page 8, 24)

[Menze et al., 2015] Menze, M., Heipke, C., and Geiger, A. (2015). Discrete optimization for optical flow. In *Proc. German Conference on Pattern Recognition (GCPR)*. (page 126)

[Milford et al., 2015] Milford, M., Kim, H., Leutenegger, S., and Davison, A. (2015). Towards visual slam with event-based cameras. In *The Problem of Mobile Sensors Workshop in conjunction with RSS*. (page 96)

[Mueggler et al., 2015] Mueggler, E., Gallego, G., and Scaramuzza, D. (2015). Continuous-time trajectory estimation for event-based vision sensors. In *Robotics: Science and Systems*. (page 95, 96)

[Mueggler et al., 2014] Mueggler, E., Huber, B., and Scaramuzza, D. (2014). Event-based, 6-dof pose tracking for high-speed maneuvers. In *International Conference on Intelligent Robots and Systems*. (page 95, 96)

[Murota, 2003] Murota, K. (2003). *Discrete Convex Analysis*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, USA. (page 28)

[Nesterov, 2004] Nesterov, I. (2004). *Introductory Lectures on Convex Optimization: A Basic Course*. Mathematics and its applications. Kluwer Academic Publishers. (page 8, 13, 23)

[Nesterov, 1983] Nesterov, Y. (1983). A method of solving a convex programming problem with convergence rate o(1/k2). *Soviet Mathematics Doklady*, 27(2):372–376. (page 9, 24)

[Newcombe et al., 2011] Newcombe, R., Lovegrove, S. J., and Davison, A. J. (2011). DTAM: Dense tracking and mapping in real-time. In *Proc. International Conference on Computer Vision (ICCV)*. (page 77, 84)

[Pock and Chambolle, 2011] Pock, T. and Chambolle, A. (2011). Diagonal Preconditioning for First Order Primal-dual Algorithms in Convex Optimization. In *Proc. International Conference on Computer Vision (ICCV)*. (page 89)

[Pock et al., 2010] Pock, T., Cremers, D., Bischof, H., and Chambolle, A. (2010). Global solutions of variational models with convex regularization. *SIAM Journal on Imaging Sciences*, 3(4):1122–1145. (page 9)

[Pock et al., 2008] Pock, T., Schoenemann, T., Graber, G., Bischof, H., and Cremers, D. (2008). A convex formulation of continuous multi-label problems. In *Proc. European Conference on Computer Vision (ECCV)*. (page 8)

[Pons et al., 2006] Pons, J.-P., Keriven, R., and Faugeras, O. (2006). Multi-View Stereo Reconstruction and Scene Flow Estimation with a Global Image-Based Matching Score. *International Journal of Computer Vision (IJCV)*, 72(2):179–193. (page 75)

[Prados and Faugeras, 2005] Prados, E. and Faugeras, O. (2005). A generic and provably convergent shape-from-shading method for orthographic and pinhole cameras. *International Journal of Computer Vision (IJCV)*, 65(1-2):97–125. (page 78)

[Ranftl et al., 2014] Ranftl, R., Bredies, K., and Pock, T. (2014). Non-local total generalized variation for optical flow estimation. In *Proc. European Conference on Computer Vision (ECCV)*. (page 108)

[Ranftl et al., 2012] Ranftl, R., Gehrig, S., Pock, T., and Bischof, H. (2012). Pushing the limits of stereo using variational stereo estimation. In *Proc. Intelligent Vehicles Symposium*. (page 78, 108)

[Ranftl et al., 2013] Ranftl, R., Pock, T., and Bischof, H. (2013). Minimizing tgv-based variational models with non-convex data terms. In *Proc. Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*. (page 9)

[Raposo et al., 2014] Raposo, C., Antunes, M., and Barreto, J. P. (2014). Piecewise-Planar StereoScan: Structure and Motion from Plane Primitives. In *Proc. European Conference on Computer Vision (ECCV)*. (page 77)

[Rastegari et al., 2016] Rastegari, M., Ordonez, V., Redmon, J., and Farhadi, A. (2016). Xnor-net: Imagenet classification using binary convolutional neural networks. In *Proc. European Conference on Computer Vision (ECCV)*. (page 108, 109)

[Ratner and Schechner, 2007] Ratner, N. and Schechner, Y. Y. (2007). Illumination multiplexing within fundamental limits. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 100)

[Revaud et al., 2015] Revaud, J., Weinzaepfel, P., Harchaoui, Z., and Schmid, C. (2015). EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 125)

[Rockafellar, 1970] Rockafellar, R. T. (1970). *Convex analysis*. Princeton Mathematical Series. Princeton University Press. (page 13)

[Rudin et al., 1992] Rudin, L. I., Osher, S., and Fatemi, E. (1992). Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1):259 – 268. (page 100)

[Shekhovtsov et al., 2008] Shekhovtsov, A., Kovtun, I., and Hlavác, V. (2008). Efficient MRF deformation model for non-rigid image matching. *Computer Vision and Image Understanding*, 112(1):91–99. (page 108, 115, 116)

[Shekhovtsov et al., 2016] Shekhovtsov, A., Reinbacher, C., Graber, G., and Pock, T. (2016). Solving dense image matching in real-time using discrete-continuous optimization. In *Proc. Computer Vision Winter Workshop (CVWW).* (page 7, 35, 108, 115, 117)

[Simo-Serra et al., 2015] Simo-Serra, E., Trulls, E., Ferraz, L., Kokkinos, I., Fua, P., and Moreno-Noguer, F. (2015). Discriminative learning of deep convolutional feature point descriptors. In *Proc. International Conference on Computer Vision (ICCV).* (page 110)

[Sormann et al., 2007] Sormann, M., Zach, C., Bauer, J., Karner, K., and Bishof, H. (2007). Watertight Multi-View Reconstruction Based On Volumetric Graph-Cuts. In *Proc. Scandinavian Conference on Image Analysis.* (page 76)

[Spivak, 1999] Spivak, M. (1999). *A comprehensive introduction to differential geometry.* Number 1 in A Comprehensive Introduction to Differential Geometry. Publish or Perish, Houston, 3rd edition. (page 35, 68)

[Stam, 2003] Stam, J. (2003). Flows on surfaces of arbitrary topology. *ACM Transactions on Graphics*, 22(3):724–731. (page 98)

[Steidl and Teuber, 2010] Steidl, G. and Teuber, T. (2010). Removing multiplicative noise by douglas-rachford splitting methods. *Journal of Mathematical Imaging and Vision*, 36(2):168–184. (page 101)

[Strecha et al., 2008] Strecha, C., von Hansen, W., Van Gool, L., Fua, P., and Thoennessen, U. (2008). On benchmarking camera calibration and multi-view stereo for high resolution imagery. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR).* (page 92)

[Stühmer et al., 2010] Stühmer, J., Gumhold, S., and Cremers, D. (2010). Real-Time Dense Geometry from a Handheld Camera. In *Pattern Recognition (Proc. DAGM).* (page 77, 84, 85)

[Taniai et al., 2014] Taniai, T., Matsushita, Y., and Naemura, T. (2014). Graph Cut Based Continuous Stereo Matching Using Locally Shared Labels. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR).* (page 78)

[Tanskanen et al., 2013] Tanskanen, P., Kolev, K., Meier, L., Camposeco, F., Sauer, O., and Pollefeys, M. (2013). Live Metric 3D Reconstruction on Mobile Phones. In *Proc. International Conference on Computer Vision (ICCV).* (page 76)

[Tarlow et al., 2011] Tarlow, D., Batra, D., Kohli, P., and Kolmogorov, V. (2011). Dynamic tree block coordinate ascent. In *Proc. International Conference on Machine Learning (ICML).* (page 7)

[Tola et al., 2011] Tola, E., Strecha, C., and Fua, P. (2011). Efficient large-scale multi-view stereo for ultra high-resolution image sets. *Machine Vision and Applications*, 23(5):903–920. (page 76)

[Topkis, 1978] Topkis, D. M. (1978). Minimizing a submodular function on a lattice. *Operations Research*, 26(2):305–321. (page 28)

[Trzcinski et al., 2013] Trzcinski, T., Christoudias, M., Fua, P., and Lepetit, V. (2013). Boosting binary keypoint descriptors. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 108)

[Tyleček and Šará, 2010] Tyleček, R. and Šará, R. (2010). Refinement of Surface Mesh for Accurate Multi-View Reconstruction. *International Journal of Virtual Reality*, 9(1):45–54. (page 75)

[Unger et al., 2012] Unger, M., Werlberger, M., Pock, T., and Bischof, H. (2012). Joint motion estimation and segmentation of complex scenes with label costs and occlusion modeling. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 128)

[Valkonen, 2014] Valkonen, T. (2014). A primal–dual hybrid gradient method for nonlinear operators with applications to mri. *Inverse Problems*, 30(5):055012. (page 128)

[Wainwright et al., 2005] Wainwright, M. J., Jaakkola, T. S., and Willsky, A. S. (2005). Map estimation via agreement on (hyper)trees: message-passing and linear programming. *Transactions on Information Theory*, 51(11):3697–3717. (page 6, 34, 115, 116)

[Wainwright and Jordan, 2008] Wainwright, M. J. and Jordan, M. I. (2008). Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305. (page 31)

[Wang and Zheng, 2008] Wang, Z.-F. and Zheng, Z.-G. (2008). A region based stereo matching algorithm using cooperative optimization. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 77)

[Weikersdorfer et al., 2013] Weikersdorfer, D., Hoffmann, R., and Conradt, J. (2013). Simultaneous localization and mapping for event-based vision systems. In *International Conference on Computer Vision Systems*. (page 95, 96)

[Wendel et al., 2012] Wendel, A., Maurer, M., Graber, G., Pock, T., and Bischof, H. (2012). Dense reconstruction on-the-fly. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 77, 84)

[Werner, 2007] Werner, D. (2007). *Funktionalanalysis*. Springer-Lehrbuch. Springer Berlin Heidelberg, 6th edition. (page 46)

[Wiesmann et al., 2012] Wiesmann, G., Schraml, S., Litzenberger, M., Belbachir, A. N., Hofstätter, M., and Bartolozzi, C. (2012). Event-driven embodied system for feature extraction and object recognition in robotic applications. In *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. (page 94)

[Woodford et al., 2009] Woodford, O., Torr, P., Reid, I., and Fitzgibbon, A. (2009). Global Stereo Reconstruction Under Second-Order Smoothness Priors. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 31(12):2115–2128. (page 78)

[Xu et al., 2017] Xu, J., Ranftl, R., and Koltun, V. (2017). Accurate Optical Flow via Direct Cost Volume Processing. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 108, 111, 125, 126)

[Yashtini and Kang, 2015] Yashtini, M. and Kang, S. H. (2015). Alternating direction method of multiplier for euler's elastica-based denoising. In Aujol, J.-F., Nikolova, M., and Papadakis, N., editors, *Proc. Conference on Scale Space and Variational Methods in Computer Vision (SSVM)*. (page 128)

[Zabih and Woodfill, 1994] Zabih, R. and Woodfill, J. (1994). Non-parametric local transforms for computing visual correspondence. In *Proc. European Conference on Computer Vision (ECCV)*. (page 3)

[Zagoruyko and Komodakis, 2015] Zagoruyko, S. and Komodakis, N. (2015). Learning to compare image patches via convolutional neural networks. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 128)

[Zaharescu et al., 2011] Zaharescu, A., Boyer, E., and Horaud, R. (2011). Topology-adaptive mesh deformation for surface evolution, morphing, and multiview reconstruction. *Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 33(4):823–37. (page 75)

[Žbontar and LeCun, 2015] Žbontar, J. and LeCun, Y. (2015). Computing the stereo matching cost with a convolutional neural network. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*. (page 108, 113)

[Zhang et al., 2014] Zhang, C., Li, Z., Cai, R., Chao, H., and Rui, Y. (2014). As-Rigid-As-Possible Stereo under Second Order Smoothness Priors. In *Proc. European Conference on Computer Vision (ECCV)*. (page 78)

[Zhang and Chen, 2016] Zhang, J. and Chen, K. (2016). A new augmented lagrangian primal dual algorithm for elastica regularization. *Journal of Algorithms & Computational Technology*, 10(4):325–338. (page 128)

[Živný et al., 2014] Živný, S., Werner, T., and Průša, D. a. (2014). The power of lp relaxation for map inference. In Nowozin, S., Gehler, P. V., Jancsary, J., and Lampert,

C. H., editors, *Advanced Structured Prediction*, pages 19–42. The MIT Press, Cambridge, USA. (page 31)