Thèse n° 7086

Learning stereo reconstruction with deep neural networks

Présentée le 28 février 2020

à la Faculté des sciences et techniques de l'ingénieur
Laboratoire de l'IDIAP
Programme doctoral en génie électrique

pour l'obtention du grade de Docteur ès Sciences

par

Stepan TULYAKOV

Acceptée sur proposition du jury

Prof. P. Frossard, président du jury
Dr. F. Fleuret, Dr A. Ivanov, directeurs de thèse
Prof. R. Sznitman, rapporteur
Dr J. D. Wegner, rapporteur
Prof. A. Alahi, rapporteur

École
polytechnique
fédérale
de Lausanne

2020

Acknowledgements

First, I would like to thank my advisor, professor *François Fleuret* for accepting me as his student and allowing me to follow my interest in Machine Learning. His enthusiastic supporting and guidance allowed me to complete my studies. I believe I learned a lot from him on personal and professional levels. Also, I am grateful to my co-advisor professor *Anton Ivanov* for bringing me to Space Center, providing me with a unique opportunity to work with beautiful Mars images and guiding me through my first years in Switzerland.

I would like to thank the members of the thesis committee: professors *Alexandre Alahi*, *Raphael Sznitman*, doctor *Jan Dirk Wegner* and the jury president, professor *Pascal Frossard* for their interest in my work and insightful comments.

I am very grateful to my colleagues from the University of Bern, in particular, professor *Nicolas Thomas*, doctors *Antoine Pommerol*, *Mattew Read* and *Miguel Almeida* for numerous useful discussion about CaSSIS and warm welcome in Bern. Also, I thank my colleagues from Space Center, and especially my office mates *Candice Norhadian* and *Michaël Juillard* for always being there for me.

And, of course, I thank my parents *Valera Tulyakov* and *Irina Tulyakova* for their encouragement and support and my wife *Hyunjung Kim* for being with me through this journey.

Finally, I thank *NCCR PlanetS* and *CaSSIS project* of the University of Bern funded through the Swiss Space Office via ESA's PRODEX program for supporting my research.

Abstract (French)

La reconstruction stéréoscopique consiste à retrouver une structure 3D à partir d'une paire d'images de la scène, obtenues depuis des angles différents. Ce problème est étudié depuis des décennies, et de nombreuses méthodes ont été développées avec succès.

Le désavantage principal de ces méthodes est qu'elles utilisent généralement une seule source d'information à propos de la profondeur, telle que la parallaxe, le flou de défocalisation ou l'ombrage, et pour cette raison elles ne sont pas aussi robustes que le système visuel humain qui repose sur un éventail d'indices monoculaires et binoculaires. La raison principale en est qu'il est trop difficile de concevoir manuellement un modèle utilisant de multiples sources d'informations à propos de la profondeur. Dans ce travail, nous abordons ce problème en nous concentrant sur les méthodes stéréoscopiques utilisant de l'apprentissage profond qui peuvent, à partir des données accompagnées de la vérité terrain concernant la profondeur, directement produire un modèle combinant des sources d'informations multiples.

Toutefois, la complexité des méthodes d'apprentissage profond exige des ensembles d'apprentissage de très grandes tailles, accompagnés de la vérité terrain, ce qui est très souvent difficile ou coûteux à obtenir. De plus, même quand la vérité terrain est disponible, elle est très souvent contaminées avec des bruits qui réduisent l'efficacité de l'apprentissage supervisé. Dans le Chapitre 3 de ce manuscrit, nous démontrons qu'il est possible d'atténuer ce problème en utilisant un apprentissage faiblement supervisé qui utilise des contraintes géométriques du problème au lieu de la profondeur réelle.

Outre un ensemble d'apprentissage de grande taille, les méthodes stéréoscopiques profondes ne sont pas utilisables pour autant d'applications que les méthodes traditionnelles. Elles demandent une capacité mémoire importante, et leur plage de disparité est fixée au moment de l'apprentissage. Dans le Chapitre 4 du présent travail, nous abordons ces deux aspects en introduisant une architecture de réseau innovatrice avec un goulot dans la représentation, capable de traiter de grandes images en utilisant plus de contexte, et un estimateur qui rend le réseau moins sensible aux ambiguïtés dans les correspondances stéréo, et applicable à de nouvelles plages de disparité sans nécessiter de ré-entraînement.

Parce que les méthodes d'apprentissage profond sont capables de découvrir des indications de profondeur dans les données d'apprentissage, elles peuvent être adaptées à de nouvelles modalités sans faire des modification importante. Dans le Chapitre 5 de la présente étude, nous démontrons que notre méthode, conçue pour une caméra traditionnelle, peut être utilisée avec une nouvelle technologie de caméras, dites "à évenements" disposant d'une plage plus dynamique, de latence plus faible, et de consommation d'énergie réduite. Au lieu de l'intensité d'échantillonnage de tous les pixels à une fréquence fixée, cette caméra capture et transmet d'une manière asynchrone tous les événements des changements d'intensités de pixels. Pour adapter notre méthode à cette nouvelle forme d'imagerie, nous proposons un module d'intégration de séquences d'événements qui regroupe initialement les données dans le temps, localement et de manière continue à l'aide d'une couche entièrement connectée, puis spatialement de manière discrète.

Une application très intéressante de la reconstruction stéréoscopique est la reconstruction de la topographie de la surface d'une planète à l'aide d'images acquises par un satellite. Dans le Chapitre 6 du présent travail, nous décrivons un procédé de calibration géométrique, ainsi que les outils de la reconstruction mosaïque et stéréoscopique que nous avons développés dans le cadre du projet doctoral pour le *Color and Stereo Surface Imaging System* embarqué à bord du *Trace Gas Orbiter* de l'ESA, orbitant autour de Mars. Pour l'étalonnage, nous proposons une méthode innovatrice qui se base sur les images de champs d'étoiles parce que des longueurs focales importantes, et la distorsion optique complexe de l'instrument, interdisent l'utilisation des méthodes classiques. Les résultats scientifiques et pratiques de ce travail sont largement utilisés par la communauté scientifique.

Mots clés: 3D, stéréo, faiblement supervisé, entropie croisée sous-pixel, MAP sous-pixel, caméra événementielle, calibration géométrique, CaSSIS.

Abstract (English)

Stereo reconstruction is a problem of recovering a 3d structure of a scene from a pair of images of the scene, acquired from different viewpoints. It has been investigated for decades and many successful methods were developed.

The main drawback of these methods, is that they typically utilize a single depth cue, such as parallax, defocus blur or shading, and thus are not as robust as a human visual system that simultaneously relies on a range of monocular and binocular cues. This is mainly because it is hard to manually design a model, accounting for multiple depth cues. In this work, we address this problem by focusing on deep learning-based stereo methods that can discover a model for multiple depth cues directly from training data with ground truth depth.

The complexity of deep learning-based methods, however, requires very large training sets with ground truth depth, which is often hard or costly to collect. Furthermore, even when training data is available it is often contaminated with noise, which reduces the effectiveness of supervised learning. In this work, in Chapter 3 we show that it is possible to alleviate this problem by using weakly supervised learning, that utilizes geometric constraints of the problem instead of ground truth depth.

Besides the large training set requirement, deep stereo methods are not as application-friendly as traditional methods. They have a large memory footprint and their disparity range is fixed at training time. For some applications, such as satellite stereo imagery, these are serious problems since satellite images are very large, often reaching tens of megapixels, and have a variable baseline, depending on a time difference between stereo images acquisition. In this work, in Chapter 4 we address these problems by introducing a novel network architecture with a bottleneck, capable of processing large images and utilizing more context, and an estimator that makes the network less sensitive to stereo matching ambiguities and applicable to any disparity range without re-training.

Because deep learning-based methods discover depth cues directly from training data, they can be adapted to new data modalities without large modifications. In this work, in Chapter 5 we show that our method, developed for a conventional frame-based camera, can be used with a novel event-based camera, that has a higher dynamic range, smaller latency, and low power consumption. Instead of sampling intensity of all pixels with a fixed frequency, this camera asynchronously reports events of significant pixel intensity changes. To adopt our method to this new data modality, we propose a novel event sequence embedding module, that firstly aggregates information locally, across time, using a novel fully-connected layer for an irregularly sampled continuous domain, and then across discrete spatial domain.

One interesting application of stereo is a reconstruction of a planet's surface topography from satellite stereo images. In this work, in Chapter 6 we describe a geometric calibration method, as well as mosaicing and stereo reconstruction tools that we developed in the framework of the doctoral project for Color and Stereo Surface Imaging System onboard of ESA's Trace Gas Orbiter, orbiting Mars. For the calibration, we propose a novel method, relying on starfield images because large focal lengths and complex optical distortion of the instrument forbid using standard methods. Scientific and practical results of this work are widely used by a scientific community.

Keywords: stereo, deep learning, weakly supervised, efficient, sub-pixel cross-entropy, sub-pixel MAP, event-based camera, geometric calibration, CaSSIS.

Contents

Acknowledgements i

Abstract (French) iii

Abstract (English) v

List of Figures xi

List of Tables xiii

1 Introduction 1

- 1.1 Research area . . . 1
- 1.2 Motivation . . . 2

2 Background 5

- 2.1 Geometry . . . 6
  - 2.1.1 Homogeneous coordinates . . . 6
  - 2.1.2 Geometric camera model . . . 7
  - 2.1.3 Epipolar geometry . . . 8
  - 2.1.4 Camera calibration . . . 9
  - 2.1.5 Stereo triangulation . . . 10
  - 2.1.6 Stereo rectification . . . 10
- 2.2 Traditional stereo . . . 12
  - 2.2.1 Bayesian inference . . . 12
  - 2.2.2 Image model . . . 12
  - 2.2.3 Priors . . . 13
  - 2.2.4 Likelihood . . . 14
  - 2.2.5 Optimization . . . 15
- 2.3 Deep stereo matching . . . 15
  - 2.3.1 Hybrid approaches . . . 15
  - 2.3.2 End-to-end approaches . . . 16
- 2.4 Weakly supervised methods . . . 17
  - 2.4.1 Types of weak supervision . . . 17
  - 2.4.2 Deep weakly supervised stereo . . . 18

2.5 Stereo datasets 20
2.6 Accuracy measures 23

I Research part 25

3 Weakly supervised learning of deep patch-matching cost **27**
3.1 Introduction 28
3.2 Method 28
3.2.1 Problem formulation 28
3.2.2 Stereo matching constraints 29
3.2.3 Weakly supervised methods 30
3.3 Experiments 34
3.3.1 Parameters tuning 35
3.3.2 Comparison of the proposed methods 36
3.3.3 Comparison with the supervised method 36
3.3.4 Benchmarking 37
3.3.5 What does the deep matching cost network learn? 37
3.3.6 Weakly supervised domain adaptation 39
3.3.7 Training with a noisy ground truth 40
3.4 Conclusion 41

4 Applications-friendly deep stereo **43**
4.1 Introduction 44
4.2 Method 45
4.2.1 Matching module 46
4.2.2 Sub-pixel maximum a posteriori estimator 48
4.2.3 Sub-pixel cross-entropy loss 49
4.3 Experiments 50
4.3.1 Training on full-size images 50
4.3.2 Effectiveness of the proposed estimator 51
4.3.3 Effectiveness of the proposed loss 53
4.3.4 Benchmarking 53
4.4 Conclusion 53

5 Dense deep event-based stereo **57**
5.1 Introduction 58
5.2 Background 59
5.2.1 Event-based stereo 60
5.2.2 Deep learning with event sequences 61
5.3 Method 62
5.3.1 Events sequence embedding 63
5.4 Experiments 64
5.4.1 Dataset 67

5.4.2 Importance of spatial and temporal context 67
5.4.3 Effectiveness of the temporal aggregation 68
5.4.4 Benchmarking 69
5.4.5 What does continuous fully-connected layer learn? 70
5.5 Conclusion 73

II Applied part **75**

6 Calibration of a satellite stereo system **77**
6.1 Introduction 78
6.2 Background 79
6.2.1 Camera model 81
6.2.2 Calibration using starfield images 82
6.2.3 Space image processing tools 83
6.3 Optical distortion model selection 83
6.4 On-ground calibration 83
6.5 In-flight calibration 85
6.5.1 Method 85
6.5.2 Results 87
6.6 Color mosaicing 91
6.6.1 Method 91
6.6.2 Results 92
6.7 Stereo reconstruction 94
6.7.1 Method 94
6.7.2 Results 96
6.8 Conclusion 98

7 Conclusions **99**
7.1 Summary 99
7.2 Future directions 100

Bibliography **120**

Curriculum Vitae **121**

List of Figures

2.1 Explanation of the pinhole camera model 8
2.2 Explanation of the epipolar geometry 9
2.3 Explanation of the stereo triangulation 9
2.4 Explanation of the stereo rectification 11
2.5 Distance to a 3d point for rectified stereo pair 11
2.6 Typical examples from the Middlebury dataset 21
2.7 Typical examples from the KITTI dataset 21
2.8 Typical examples from the FlyingThings3d dataset 22

3.1 Visualization of the stereo matching constraints 30
3.2 The problem of the Multi-Instance Learning method 32
3.3 The architecture of the deep matching cost network 35
3.4 Matching cost matrices before and after the training 39
3.5 Failure cases of the proposed deep matching cost 40

4.1 Functional structure of the proposed network 45
4.2 Explanation of the matching module 48
4.3 Disadvantages of the SoftArgmin 48
4.4 Target distribution of the proposed loss 49
4.5 Disparity errors with the proposed estimator and the baseline 52
4.6 Comparison of the convergence speed with L^1 and sub-pixel cross-entropy . . 52

5.1 Working principles of a conventional and an event-based sensor 59
5.2 Functional structure of the event-based stereo network 62
5.3 Illustration of the event queue 63
5.4 Comparison of the continuous fully-connected layer to the baseline 65
5.5 Explanation of the continuous fully-connected layer 65
5.6 The continuous kernels before and after the custom initialization 66
5.7 Visualization of kernel network output before and after the training 71
5.8 Qualitative comparison with the recent event-based stereo methods 72

6.1 CaSSIS sensor outline and specification 80
6.2 Color acquisition capability of the CaSSIS 80

6.3 Stereo acquisition capability of the CaSSIS 80
6.4 Visualization of CaSSIS coordinate frames 81
6.5 Distortion fields of various optical distortion models fitted to the simulated data 84
6.6 Image of the calibration chart overlayed with the grid 85
6.7 On-ground calibration setting 85
6.8 Workflow of in-flight calibration 86
6.9 Position of stars detected in the training set on the sensor 88
6.10 Residual reprojection error after the first and the eleventh BA iterations 89
6.11 Distortion fields estimated from starfield images and optical simulation 89
6.12 Residual reprojection error after taking into account the optical distortion . . . 90
6.13 Workflow of the proposed color mosaicing method 91
6.14 Examples of color mosaics reconstructed using the proposed method 93
6.15 Close-ups of the color mosaics 94
6.16 Pixel-wise bands misalignment in the reconstructed color mosaics 95
6.17 Hillshade images synthesized from the CaSSIS DTMs 97

List of Tables

2.1 Summary of stereo datasets 20

3.1 Parameters tuning for the proposed method 35
3.2 Comparison of the proposed weakly supervised methods 36
3.3 Comparison of the proposed method with the fully-supervised baseline 36
3.4 Benchmarking results on the Middlebury dataset 38
3.5 Benchmarking results on the KITTI datasets 38
3.6 Generalization error across the datasets 40
3.7 Training in the presence of ground truth noise 41

4.1 The architecture of the proposed network 47
4.2 Summary of training settings for the FlyingThings3D and the KITTI datasets . . 51
4.3 Disparity error as a function of training and test patch sizes 51
4.4 Effectiveness of the proposed estimator compared to the baseline 52
4.5 Effectiveness of the proposed loss compared to the baseline 53
4.6 Benchmarking results on the FlyingThings3d dataset 54
4.7 Benchmarking results on the KITTI 2012 and KITTI 2015 datasets 55

5.1 Advantages and drawbacks of an event-based sensor 60
5.2 The architecture of the kernel network 65
5.3 Optimal parameters setting 66
5.4 Summary of the splits of the Indoor Flying dataset 68
5.5 Importance of temporal context in the presence of large and small spatial context 68
5.6 Effectiveness of the proposed temporal aggregation method 69
5.7 Benchmarking results on the Indoor Flying dataset 71

6.1 Summary of the starfield datasets 87
6.2 Estimated parameters of the Rational distortion model 90
6.3 Reprojection error on the validation set 90
6.4 Estimated rotation matrices 90
6.5 The CaSSIS observations that we use for the qualitative analysis of color mosaics 92
6.6 Average misalignment between bands in the color mosaics 92
6.7 The CaSSIS observations that we use for analysis of DTMs quality 96

6.8 HiRISE DTMs that we use as ground truth for analysis of CaSSIS DTMs 96
6.9 Average absolute vertical errors of the CaSSIS DTMs 98

1 Introduction

The main objective of this chapter is to introduce a reader to the research area and provide a motivation for this work. Detailed explanations of the proposed methods will be given in the following chapters.

1.1 Research area

This dissertation combines recent advances in Computer Vision and Deep Learning fields to develop novel stereo reconstruction methods. The *Computer Vision* is an area of Artificial Intelligence that extracts information about the real world, such as scene structure and semantics, from visual observations, such as videos and images. This is a so-called *inverse problem* because it attempts to calculate causal factors that produce an observation from the observation. The corresponding direct problem is addressed in *Computer Graphics* which renders visual observations from models of the real-world objects.

Computer vision problems can be divided into *low-level problems*, which deal with physical properties and *high-level problems*, which deal with semantics. The low-level problems, for example, include edge detection [1], denoising [2], color processing [3], optic flow [4] and stereo reconstruction, addressed in this work.

Goal of the *stereo reconstruction* is to estimate a 3d structure of a scene from two images of the scene acquired from different viewpoints. In these images, corresponding points appear shifted and from the extent of this shift, we can compute the distance to the corresponding physical point. The problem has applications in robotics [5], medical imaging [6], remote sensing [7], virtual reality, 3d graphics, and computational photography [8, 9].

Most of the problems in the Computer Vision, including the stereo reconstruction, are very hard for two reasons. The first reason is that most of them are *ill-posed*, *i.e.* cannot be uniquely solved without additional assumptions about the solution. Therefore, all computer vision methods must rely on domain-specific prior knowledge which is often impossible to hard-code. The second reason, often referred to as *curse of dimensionality*, is a high dimensionality of input and outputs spaces of the problems. In such case, it is very hard to find meaningful and robust features related to the solution. Fortunately, Machine Learning and, in particular, Deep Learning address both problems.

Machine Learning (ML) is an area of Artificial Intelligence, that allows inferring algorithms and patterns directly from *training data* without explicitly hard-coding them. The training data usually consists of examples of input and desired output, coupled with a *loss function* that allows comparing the output produced by a method to the desired output. *Deep Learning* is a successful sub-field of Machine Learning, that relies on the models inspired by biological neural networks with multiple layers of neurons. To date, Deep Learning was successfully applied to many computer vision problems, such as object detection and segmentation [10], pose estimation [11], reinforcement learning [12], image generation [13], automatic image captioning [14] and visual question answering [15]. However, there were few prior attempts to apply it to the stereo reconstruction.

1.2 Motivation

The main motivation of Research Part of this dissertation is to advance stereo reconstruction methods, while considering their future application in satellite stereo imagery.

The stereo reconstruction problem has been investigated for decades and many successful methods were developed over the years. However, these methods are still not as robust and accurate as a human visual system. The human visual system uses multiple monocular and binocular depth cues for inferring the distance to objects. The *binocular cues*, rely on our ability to observe the real world with two eyes and they are strongest and most reliable. They, for example, include *stereopsis cue*, which is a displacement of physically corresponding points in images acquired from different viewpoints. However, in addition to the binocular cues, our visual system simultaneously uses multiple subtle *monocular cues*, such as, for example, defocus blur, shadows, textures gradients, prior knowledge about object sizes. This gives it robustness unattainable by current stereo reconstruction methods engineered by a human. In contrast, the engineered stereo reconstruction methods mostly rely on a single depth cue, typically on the stereopsis, shading [16], defocus blur [17], and texture gradients [18]. This is mainly because it is hard to manually design models for multiple depth cues. We believe, that this problem can be addressed by using deep learning-based methods, that can discover multiple depth cues directly from training data with ground truth depth and learn how to optimally combine them. Encouraged by the first success [19] of deep learning-based methods in stereo reconstruction, we decided to focus on them in this dissertation.

The complexity of deep learning models, however, requires large training sets with ground truth depth, which are hard or costly to collect. Moreover, for some applications, such as satellite stereo imagery of Mars, collecting such training sets is technically impossible. Besides, even when the labeled training set is available, the ground truth is usually produced automatically from a depth sensor, such as LiDAR, and thus often contains noise that reduces the effectiveness of supervised learning [20]. The current solution to this problem is to use large synthetic training sets [21] with ground truth, created using computer graphics. The synthetic training data is, however, different from target domain data. This *domain gap* manifests in a large performance drop when the network is deployed in the target domain without fine-tuning. We believe that the large labeled training set requirement can be alleviated using *weakly supervised learning* that utilizes geometric constraints instead of ground truth depth, and explore this idea in Chapter 3 of this work.

Besides requiring a large amount of training data with ground truth, deep learning-based stereo methods are not as application-friendly as traditional methods. Their biggest disadvantages are a large memory footprint and a disparity range fixed at training time. For some applications, such as satellite stereo imagery, these are serious problems since satellite images are very large, often reaching tens of megapixels, and have a variable baseline, depending on a time difference between stereo images acquisition. We believe that deep stereo methods can be made more application-friendly by using domain knowledge during network design and experiment with this idea in Chapter 4 of this dissertation.

Because deep learning methods discover depth cues directly from training data, they can be easily adapted to novel data modalities without serious modifications. This is intriguing since many machine vision applications could benefit from switching to more advanced sensors, which do not suffer from high power consumption, data rate, latency, and low dynamic range. One interesting alternative to a conventional sensor is a novel *event-based sensor*, mimicking a biological retina and reporting only asynchronous events of significant pixel intensity changes. In Chapter 5 we explore the possibility of applying the deep stereo method, developed in Chapter 4, to an event-based stereo system.

The main motivation of the Applied Part of this work is to develop geometric calibration, stereo reconstruction and color mosaicing tools for *Color and Stereo Surface Imaging System (CaSSIS)* onboard of ESA's ExoMars *Trace Gas Orbiter (TGO)*, orbiting Mars. To prepare high-quality color mosaics and *Digital Terrain Models (DTMs)* from raw CaSSIS data, we need precise geometric parameters of the camera, such as its focal length, optical distortion model and rotation relative to the spacecraft frame. While the nominal values of these parameters are known from the technical specifications, their actual values have to be measured in a laboratory and validated during commissioning. This procedure is known as a *geometric camera calibration*. In our case, large focal length and complex optical distortion of the telescope forbid using standard calibration methods relying on images of a calibration chart. Therefore, in Chapter 6 of this work, we explore the possibility of calibrating the telescope using images of starfields.

2 Background

Contents

2.1 **Geometry** . . . 6
  2.1.1 Homogeneous coordinates . . . 6
  2.1.2 Geometric camera model . . . 7
  2.1.3 Epipolar geometry . . . 8
  2.1.4 Camera calibration . . . 9
  2.1.5 Stereo triangulation . . . 10
  2.1.6 Stereo rectification . . . 10
2.2 **Traditional stereo** . . . 12
  2.2.1 Bayesian inference . . . 12
  2.2.2 Image model . . . 12
  2.2.3 Priors . . . 13
  2.2.4 Likelihood . . . 14
  2.2.5 Optimization . . . 15
2.3 **Deep stereo matching** . . . 15
  2.3.1 Hybrid approaches . . . 15
  2.3.2 End-to-end approaches . . . 16
2.4 **Weakly supervised methods** . . . 17
  2.4.1 Types of weak supervision . . . 17
  2.4.2 Deep weakly supervised stereo . . . 18
2.5 **Stereo datasets** . . . 20
2.6 **Accuracy measures** . . . 23

The main objective of this chapter is to introduce the stereo reconstruction problem and review related works. The stereo reconstruction problem is to estimate a 3d structure of a scene from two images of the scene acquired from different viewpoints. The problem has distinctive geometric and algorithmic components.

The geometric component comes to play at the beginning and the end of the stereo reconstruction. In the beginning, it is used for *stereo rectification*, which aligns stereo images and simplifies a subsequent search for image correspondences, and in the end, it is used for *stereo triangulation*, which infers the distances to image points from established image correspondences. We briefly review the geometry of the stereo reconstruction in § 2.1 of this chapter.

The algorithmic part consists in finding a disparity value for every pixel in arbitrary selected *reference stereo image*. This disparity value is inverse proportional to the distance to the corresponding physical point. For most of the physical points, the disparity is simply a horizontal shift between projections of a point in rectified stereo images. For these points, the algorithmic part simply consists in finding physically matching points in the stereo images, *i.e. stereo matching*. However, this is not true for *occluded points*, which appear only in one of the stereo images. The stereo matching is an ill-posed problem that has been investigated for decades and which is also the main focus of this dissertation. In this chapter, in § 2.2, we briefly review traditional, non-learning based approaches to the stereo matching. Then, in § 2.2 we overview modern deep learning-based approaches and in § 3.2.3 describe weakly supervised deep learning-based stereo methods. Finally, in § 2.5 we introduce stereo datasets used in this work and in § 2.6 discuss measures of stereo reconstruction accuracy.

2.1 Geometry

In this section, we briefly review the geometry of a stereo imaging system. First, in § 2.1.1 we introduce homogeneous coordinates, then, in § 2.1.2 summarize the geometric model of a single camera, in § 2.1.3 review the geometry of a stereo system and in § 2.1.4 describe how to estimate parameters of this system using a geometric calibration. Next, in § 2.1.5 we describe stereo triangulation, that given coordinates of projections of a point in two stereo images computes the distance to the corresponding physical point. Finally, in § 2.1.6 we discuss stereo rectification that greatly simplifies the search for image correspondences and the stereo triangulation. For further details about the geometry of a stereo system please refer to [22].

2.1.1 Homogeneous coordinates

Homogeneous coordinates is a convenient representation of point coordinates that allows simplifying many geometric equations. To convert a Cartesian coordinate vector to homogeneous coordinate vector, we simply add unit coordinate to the end of the Cartesian coordinate vector and multiply the resulting vector by an arbitrary constant, *i.e.* $[X, Y, Z]^T \Rightarrow [X\omega, Y\omega, Z\omega, \omega]^T$

or $[x, y]^T \Rightarrow [x\omega, y\omega, \omega]^T$, where $\omega \in \mathbb{R}$. Consequently, we can represent one point by infinitely many homogeneous coordinates. To convert a homogeneous coordinate vector to Cartesian, we divide the homogeneous coordinate vector by its last element and remove the unit coordinate from the resulting vector. Following [22], we use homogeneous coordinates representation in most of the equations. In the rare cases when we use Cartesian representation, we emphasis it by placing tilde over coordinate vectors.

2.1.2 Geometric camera model

The *geometric camera model* describes the mathematical relationship between the coordinates of a point in 3d space and its 2d coordinates in an image. It consists of *extrinsic model, intrinsic model* and *optical distortion model*, described below.

Extrinsic model [22, p155-156] describes the transformation from *world frame coordinates* (X, Y, Z) to *camera frame coordinates* (X_C, Y_C, Z_C) as follows

$$\widetilde{\mathbf{X}}_C = [\mathbf{R} \mid \mathbf{t}] \cdot \widetilde{\mathbf{X}}, \tag{2.1}$$

where $\mathbf{R}$ is a 3×3 *camera rotation matrix*, and $\mathbf{t}$ is a 3×1 *camera translation vector*, $\widetilde{\mathbf{X}}_C$ is a 3×1 vector of camera frame coordinates, and $\widetilde{\mathbf{X}}$ is a 3×1 vector of world frame coordinates. Alternatively, we can represent the rotation matrix $\mathbf{R}$ by a triplet of *Euler angels*, for example, $(\alpha_X, \alpha_Y, \alpha_Z)$. These Euler angles define tree sequential rotations around axis X, Y and Z correspondingly. Note, that we can represent the rotation matrix by several different triplets of Euler angles.

Intrinsic model [22, p153-158] describes the transformation from the 3d *camera frame coordinates* (X_C, Y_C, Z_C) to 2d *image frame coordinates* (x, y). Usually, we use a simple pinhole camera model as the intrinsic model.

The *pinhole camera model* performs a central projection of 3d points onto a projection plane as shown in Figure 2.1. The center of the projection C we call a *camera center* or an *optical center*. It is also an origin of the *camera frame coordinate system*. The projection plane Π we call an *image plane*. The axis Z_C of the camera frame we call a *principal axis* or a *principal ray* and the point P where this axis meets the image plane we call a *principal point*. The pinhole camera model performs a linear mapping in homogeneous coordinates as

$$\mathbf{x} = \mathbf{K} \cdot [\mathbf{I} \mid \mathbf{0}]\, \mathbf{X}_C, \quad \text{where} \quad \mathbf{K} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.2}$$

where $\mathbf{K}$ is a *camera calibration matrix*, $\mathbf{x}$ is a 3×1 vector of homogeneous image coordinates, $\mathbf{X}_C$ is a 4×1 vector of homogeneous camera frame coordinates, f is a *focal length* of the camera, measured in pixels, and x_0, y_0 are coordinates of a principal point in the image frame.

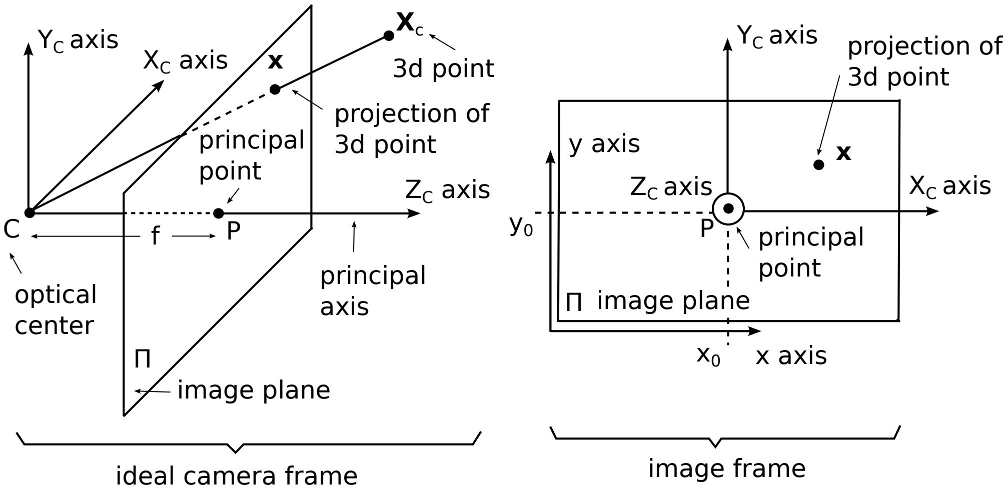


Figure 2.1 – The pinhole camera model performs a central projection of the 3d point $\mathbf{X}_C$ onto the projection plane. The center of the projection C we call a *camera center* or an *optical center*. It is also an origin of the *camera coordinates frame*. The projection plane Π we call an *image plane* and describe by the equation $Z_C = f$ in the camera frame. The axis Z_C of the camera frame we call a *principal axis* or a *principal ray* and the point P where it meets the image plane we call a *principal point*.

We can combine the intrinsic and the extrinsic models into the single equation as

$$\mathbf{x} = \mathbf{P}\cdot\mathbf{X}, \quad \mathbf{P} = \mathbf{K}\cdot[\mathbf{R}\,|\,\mathbf{t}], \tag{2.3}$$

where $\mathbf{P}$ is a 3×4 *camera projection matrix*.

Optical distortion model describes the transformation between the *image frame coordinates* (x, y) and *distorted image frame coordinates* (x_d, y_d) as $\mathbf{x}_d = f_{\text{dist}}(\mathbf{x})$ or $\mathbf{x} = f_{\text{corr}}(\mathbf{x}_d)$. The former transformation is more practical than the latter, since it can be directly used for image interpolation during the distortion correction. The optical distortion in a conventional camera we typically describe by *Radial* or *Brown-Conrady* optical distortion models. The Radial model [22, p189-193] only accounts for radially symmetric distortion, while the Brown-Conrandy [23] in addition, accounts for tangential decentering. We usually describe the distortion model in the camera frame coordinates. For example, the most common Radial model we describe as

$$\widetilde{\mathbf{X}}_{Cd} = (1 + k_1 r + k_2 r^2)\cdot\widetilde{\mathbf{X}}_C, \;\; r = \|\widetilde{\mathbf{X}}_C\|_2 \tag{2.4}$$

where k_1, k_2 are *distortion coefficients* and r is a distance to the *distortion center*.

2.1.3 Epipolar geometry

Stereo reconstruction relies on *epipolar geometry* [22, p239-262] which is simply a geometry of two pinhole cameras shown in Figure 2.2. The line connecting the camera centers C_L and

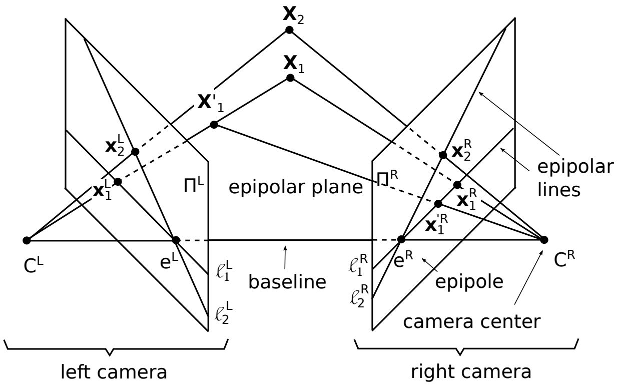


Figure 2.2 – Epipolar geometry is a geometry of two pinhole cameras. The line connecting the camera centers C_L and C_R we call a *baseline* and the points e^L and e^R where the baseline intersects the image planes we call *epipoles*. The plane containing the camera centers and a 3d point we call an *epipolar plane*. The epipolar plane intersects the image planes in the *conjugate epipolar lines* ℓ_1^L and ℓ_1^R.

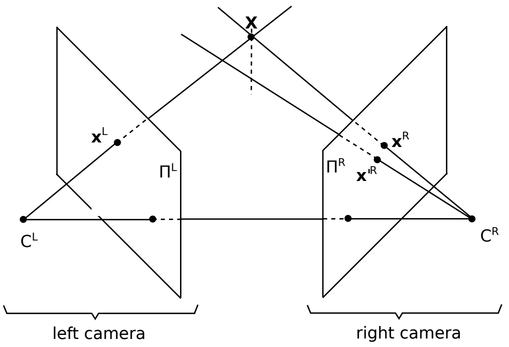


Figure 2.3 – Stereo triangulation. In practice, when we search for a point in the right view that matches the point $\mathbf{x}^L$ in the left image, instead of the exact match $\mathbf{x}^R$, we find the imprecise match $\mathbf{x}'^R$. For this imprecise match, the projection rays do not intersect.

C_R we call a *baseline* and the points e^L and e^R where the baseline intersects the image planes we call *epipoles*. The plane containing the camera centers and a 3d point we call an *epipolar plane*. The epipolar plane intersect the image planes in the *conjugate epipolar lines* ℓ_1^R and ℓ_1^L. There is an infinite number of epipolar lines in the image plane of a camera but they all intersect at the camera's epipole. Note, that the epipolar geometry of a stereo system is fully defined by geometric models of both cameras of this system.

Knowledge of epipolar geometry provides a large advantage when we search for the point in one view that corresponds to a certain point in another view because it allows reducing 2d search to 1d search along the epipolar line. For example, if in Figure 2.2 we search for a point in the right view that corresponds to the point $\mathbf{x}_1^L$ in the left view, we need to examine only points on the epipolar line ℓ_1^R in the right view, instead of all points. Notice that all the points lying on a given epipolar line in the left view conveniently correspond to the points lying on a common epipolar line in the right view.

2.1.4 Camera calibration

The process of finding camera model parameters we call *camera calibration* or *resectioning* [22, p178-194]. Typically, for the calibration we acquire several images of a *calibration chart*. The calibration chart, such as, for example, a chessboard chart, has several easily identifiable points with known 3d world frame coordinates. Therefore, the calibration image allows finding a set of corresponding world frame and image frame points $\{(\mathbf{X}_i, \mathbf{x}_i)\}_{i=1\ldots N}$. Using these points, we estimate parameters of a camera model, by minimizing average *reprojection error*, also known as a *geometric error*, which is an average distance between the predicted projection $\widehat{\mathbf{x}}$ of a 3d point $\mathbf{X}$ and its actual projection $\mathbf{x}$ as

$$\arg\min \frac{1}{N}\sum_{i}^{N} \|\mathbf{x}_i - \widehat{\mathbf{x}}_i)\|_2^2, \qquad \widehat{\mathbf{x}}_i = f_{\text{corr}}(\mathbf{P}\cdot\mathbf{X}_i) \tag{2.5}$$

This equation is non-linear due to the optical distortion correction function, and we typically solve it using *Levenberg-Marquardt* algorithm, that requires a good initialization. We can find such initialization by firstly ignoring the optical distortion and solving for the camera projection matrix using linear least squares method and factorizing the matrix. Alternatively, we can simply use for the initialization nominal camera parameters from camera specifications.

2.1.5 Stereo triangulation

The procedure of finding the distance to a physical point given coordinates of its projections in two stereo images is called *stereo triangulation*. Assume that we know 2d projections $\mathbf{x}^L$ and $\mathbf{x}^R$ of the physical point $\mathbf{X}$ as shown in Figure 2.3. In this case, we can find 3d coordinates of the point by intersecting the corresponding projection rays. In practice, however, instead of exact matching point $\mathbf{x}^R$, we find imprecise match $\mathbf{x}'^R$, for which the projection rays do not intersect. Therefore, to find 3d coordinates of the point we usually solve the following over-determined system of equations

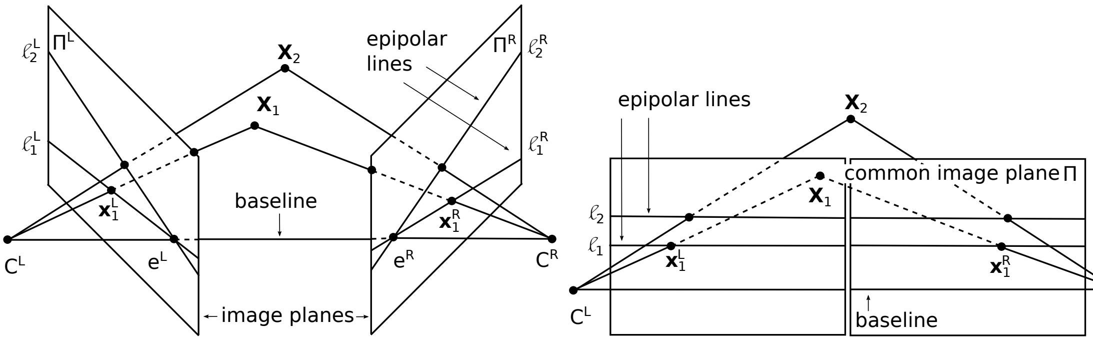
$$\left[\begin{array}{l} \mathbf{x}^L = \mathbf{P}^L\cdot\mathbf{X} \\ \mathbf{x}^R = \mathbf{P}^R\cdot\mathbf{X} \end{array}\right. \tag{2.6}$$

We can convert this system to a system of four linear equations, which we can solve using a linear least squares method.

2.1.6 Stereo rectification

Stereo rectification is a standard procedure of warping stereo images which makes the conjugate epipolar lines in these images parallel, horizontal, and vertically aligned. As a result of this procedure, we substitute the original camera models with the new ones with a common image plane parallel to the baseline of the stereo system and with epipoles moved to infinity as shown in Figure 2.4. In practice, during the stereo rectification we also correct the optical distortion and equalize the focal lengths of the cameras.

The rectification simplifies the stereo matching and the stereo triangulation. For the stereo matching, we no longer need to compute an epipolar line in an opposite view and can simply search along the horizontal line. For example, assume that we search for a point in the right image matching the point $\mathbf{x}_1^L$ in the left image. Without the rectification, this would require computing the right epipolar line ℓ_1^R and searching along this line as shown in Figure 2.4 (a). In contrast, with the rectification, we can simply search along the same horizontal line ℓ_1 in



(a) Before the rectification

(b) After the rectification

Figure 2.4 – Stereo rectification. Before the rectification (a) the epipolar lines ℓ_1^L and ℓ_2^L intersect at the epipole e^L in the left image plane Π^L and the epipolar lines ℓ_1^R and ℓ_2^R intersect at the epipole e^R in the right image plane Π^R. After the rectification (b), the cameras have a common image plane Π parallel to the baseline with parallel and horizontal epipolar lines ℓ_1 and ℓ_2.

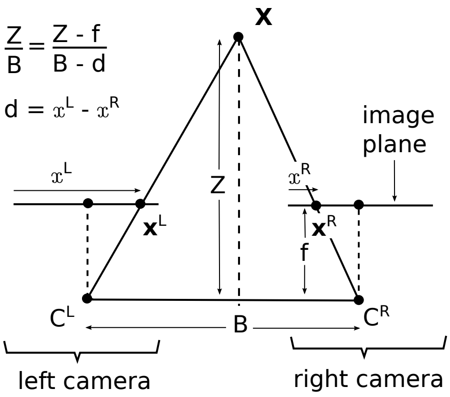


Figure 2.5 – Triangulation for a rectified stereo pair is not required. Assume that C^L and C^R are the optical centers of the left and the right cameras, B is the baseline of the stereo system and f is the focal length, equal for both cameras. There is a point $\mathbf{X}$ on the distance Z from the cameras. The left camera projects the point to the image point $\mathbf{x}^L$ and the right camera – to the image point $\mathbf{x}^R$. The projections have horizontal coordinates x^L and x^R respectively. We call the difference of the horizontal coordinates of the projections as *disparity* $d = x^L - x^R$. Note, that there is a simple geometric relationship between the distance Z and the disparity d.

the right image as shown in Figure 2.4 (b). The matching point $\mathbf{x}_1^R$ in the right image has same vertical coordinate as the point $\mathbf{x}_1^L$ in the left image, but shifted horizontal coordinate. We usually call this shift *stereo disparity*.

Moreover, as shown in Figure 2.5, for the rectified stereo pair, the triangulation is not required and the distance Z to a point is simply inverse proportional to its disparity d as

$$Z = \frac{Bf}{d}, \quad d = x^L - x^R, \tag{2.7}$$

where B is a baseline of the stereo system and f is a focal length, equal for both cameras.

2.2 Traditional stereo

In this section, we review traditional, non-learning based approaches to the stereo matching. Since stereo matching is an ill-posed problem, most of these approaches rely on *Bayesian inference* on *Markov Random Fields*, which we review in § 2.2.1 and § 2.2.2 respectively. During the Bayesian inference, the traditional approaches usually find the disparity by minimizing an *energy function* that consists of two terms: the *prior term*, that encourages solution with expected properties and the *likelihood term*, that encourages consistency with a stereo observation. We review commonly used prior and likelihood terms in § 2.2.3 and § 2.2.4 respectively. Finally, in § 2.2.5 we review methods for minimizing the energy function.

2.2.1 Bayesian inference

Early stereo matching methods [4, 24–26] relied only on information contained in a stereo observation. However, the stereo matching problem is *ill-posed* and often can not be solved without additional assumptions about the solution. The ill-posed nature of the problem manifests in *matching ambiguities, i.e.* situations when there are several equally likely solutions. To resolve the matching ambiguities, more recent stereo methods rely on *Bayesian inference* that takes into account not only a stereo observation but also prior assumptions about the solution. The former methods are now called *local methods* and the latter – *global methods.* At the core of the Bayesian inference is *Bayes Theorem*:

$$P(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R) = \frac{1}{Z} P(\mathbf{D}) \cdot P(\mathbf{I}^L, \mathbf{I}^R \mid \mathbf{D}), \tag{2.8}$$

where $P(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R)$ is a probability of the disparity $\mathbf{D}$ after observing the stereo images $\{\mathbf{I}^L, \mathbf{I}^R\}$, which is called a *posterior probability*, $P(\mathbf{D})$ is a probability of the disparity $\mathbf{D}$ without observing stereo images which is called a *prior probability*, and $P(\mathbf{I}^L, \mathbf{I}^R \mid \mathbf{D})$ is a probability of observing the stereo images $\{\mathbf{I}^L, \mathbf{I}^R\}$ given the disparity $\mathbf{D}$, which is called a *likelihood probability*, and Z is a normalization constant.

Given the posterior distribution, computed using the Bayes Theorem, one can find the optimal disparity $\widehat{\mathbf{D}}$ using a *maximum a posteriori probability (MAP)* estimator as

$$\widehat{\mathbf{D}} = \underset{\mathbf{D}}{\arg\max}\, P(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R) \tag{2.9}$$

Many stereo matching methods [27–32] use Bayesian inference and numerous heuristic methods [33–35] are inspired by it.

2.2.2 Image model

Traditional methods often represent an image by *Markov Random Field (MRF)* model. This is an undirected graphical model that assumes conditional independence of disparity in a pixel

(x, y) from disparities in far-away pixels, given disparities in the neighborhood $N(x, y)$. This neighborhood, that we call *Markov blanket*, might, for example, consist of four or eight closest pixels. The conditional independence assumption we call *Markov property* and formally write as

$$P\left(D_{y,x} \mid \left\{D_{j,i}\right\}_{(i,j)\neq(x,y)}\right) = P\left(D_{x,y} \mid \left\{D_{i,j}\right\}_{(i,j)\in N(x,y)}\right) \tag{2.10}$$

According to the *Hammersley-Clifford Theorem* and because of independence of observations in every pixel, we can factorize the prior and the likelihood distributions on MRF as

$$P(\mathbf{D}) = \prod_k \psi_{\text{prior}}\left(\left\{D_{y,x}\right\}_{(x,y)\in C_k}\right), \quad P(\mathbf{I}^L, \mathbf{I}^R \mid \mathbf{D}) = \prod_{x,y} \psi_{\text{lh}}\left(\mathbf{I}^L, \mathbf{I}^R \mid D_{x,y}\right), \tag{2.11}$$

where C is a minimal subset of MRF nodes with an edge between every pair of nodes, which is called a *clique* and $\psi_{\text{prior}}(\cdot)$ and $\psi_{\text{lh}}(\cdot)$ are likelihood and prior *clique potentials.*

The clique potentials often have an exponential form, such as $\psi(\cdot) = \exp(-\phi(\cdot))$, in which case we often substitute probability distribution functions by *energy functions* and preform minimization of a *posterior energy* $E(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R)$ instead of the maximization of the posterior probability as

$$\widehat{\mathbf{D}} = \arg\min_{\mathbf{D}} E(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R), \quad E(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R) = E_{\text{prior}}(\mathbf{D}) + E_{\text{lh}}(\mathbf{I}^L, \mathbf{I}^R \mid \mathbf{D}), \tag{2.12}$$

where $E_{\text{prior}}(\mathbf{D}) = \sum_k \phi_{\text{prior}}\left(\left\{D_{y,x}\right\}_{(x,y)\in C_k}\right)$ is a *prior energy* and $E_{\text{lh}}(\mathbf{I}^L, \mathbf{I}^R \mid \mathbf{D}) = \sum_{x,y} \phi_{\text{lh}}(\mathbf{I}^L, \mathbf{I}^R \mid D_{x,y})$ is a *likelihood energy.*

Besides MRF [27, 28], stereo matching methods might rely on other graphical models such as *Conditional Random Field (CRF)* [29, 30], *Markov Chains* [31, 32] and *Trees* [36].

All image models in traditional methods have very few parameters which we typically set empirically, based on small-scale grid search. Most of the attention in these methods we devote to finding a global minimum of the energy function.

2.2.3 Priors

The prior energy term in Equation 2.12 encodes prior assumptions about the solution.

In case of the MRF and the CRF, we typically define the prior clique potentials for a pair of pixel [37, 38] or segments [39], or segment and pixel [27, 40]. In CRFs, we can condition the clique potentials on an image, which allows, for example, adaptively attenuating disparity discontinuity penalty on texture edges [29]. The prior potentials typically encode first [37] or second [38] order smoothness, or smoothness on segment boundaries [39] or goodness of fit to a disparity plane hypothesis of a segment [41]. Besides variables for disparities, MRFs might include random variables for surface normals, occlusions, segments or objects [40–42].

The biggest limitation of MRF / CRF is that they can encode only very simple priors. Admittedly, some MRFs encode complex, even object-based priors [40, 42], however, this comes at the cost of the high complexity of the model. The big advantage of MRF / CRF is a small number of parameters and utilization of a global context.

2.2.4 Likelihood

The likelihood term in the Equation 2.12 encodes information coming from a stereo observation.

We usually compute the likelihood potentials by comparing each image patch to all its possible matches in the opposite view. This is not a trivial task since local appearance of a physical point in the two views might differ due to radiometric and geometric distortions. Therefore, we usually perform the patches comparison using *matching costs* and *descriptors*. The former ones were are more popular in the stereo matching, while the latter in matching sparse points of interest.

The *matching costs* [43, 44] are popular in stereo matching, probably due to their low computational complexity. The simplest matching cost are the sum of absolute differences (SAD), and the sum of squared differences (SSD). Zero-mean variants of these costs (ZSAD, ZSSD), as well as sum of absolute gradient differences (GSAD), are invariant to local brightness changes, which can also be achieved by combining SAD and SSD with background subtraction by mean, Laplacian of Gaussian (LoG) [45] or Bilateral filters [46]. Non-parametric matching costs, such as Rank and Census [47] are invariant to arbitrary order-preserving local intensity transformations, and matching costs such as the Mutual Information (MI) [48] explicitly model the joint intensity distribution in the two images, and are invariant to arbitrary intensity transformations. All matching costs are invariant only to radiometric distortions.

Invariant descriptors or *features* are popular for sparse point matching, and are designed to be invariant to both radiometric and geometric distortions. They all are either local histograms of oriented image gradients such as SIFT [49], SURF [50], or binary strings of local pairwise pixel comparisons such as BRIEF [51]. Although descriptors are rarely used for stereo, there are some exceptions, such as DAISY [52], which can be efficiently computed densely.

Recently, the community has moved from these fully hand-crafted descriptors to *data-driven descriptors*, directly inferred from a training data. This procedure is known as feature selection or feature extraction. The *feature selection procedure* simply selects most descriptive and uncorrelated features from a given feature pool, the *feature extraction* procedure optimally transforms original features into the new ones. Both procedures can be supervised or unsupervised. In the *unsupervised* case, there is no need in a labeled training set and features are extracted with the goal to preserve as much variance in the data as possible. In the *supervised* case, a labeled training set is required and features are extracted with the goal to preserve highly predictive features, that are most important for classes discrimination, therefore, these descriptors are called *discriminative*. There are several successful data-driven features, designed for sparse interest point matching. For example, GLOH [53] designed using unsupervised method based on PCA; BinBoost [54], VGG [55] and LDAHash [56] designed using supervised methods based on boosting, convex optimization and LDA correspondingly.

2.2.5 Optimization

For finding disparity, in the traditional approaches we have to minimize the energy function. The minimization method largely depends on the graphical model. In the case of directed graphical models without loops, such as Trees and Markov Chains, we can easily find the exact solution using *Dynamic Programming* [31, 32, 36]. In the case of undirected graphical models, such as CRF and MRF, the optimization is more complex and often approximate. For these models, we can use *Graph Cuts* [37] method, or, in more general case, *Loopy Belief Propagation* [28] or combination of *Markov Chain Monte Carlo (MCMC)* sampler and *Simulated Annealing* [57].

Overall, the high complexity of optimization is one of the biggest disadvantages of non-learning based methods. Therefore, the most successful traditional methods rely on simple graphical models, such as Markov Chains, combined with various heuristics [58].

2.3 Deep stereo matching

First deep learning-based stereo matching methods appeared in 2015 [19, 59], and their success largely inspired our work. At the beginning, deep learning replaced individual parts in legacy methods and gave rise to *hybrid methods*, which we review in § 2.3.1. The latest trend, however, consists in solving stereo matching using neural network without any post-processing. These methods are called *end-to-end methods* and we review them in § 2.3.2.

2.3.1 Hybrid approaches

In the first successful application of neural networks, researchers substituted hand-crafted stereo matching cost with deep learning-based matching costs inside a legacy stereo pipeline (often [34, 60]). Originally, researchers used similar costs for sparse point matching [21, 61–65] and later extended them to stereo reconstruction [19, 66, 67].

Standard deep patch-matching cost networks have a *Siamese architecture*, introduced in [68]. They consist of two *embedding modules* with complete weight sharing that join into a common *matching module*. Each embedding module is convolutional, it takes an image patch as input, and outputs the patch's descriptor. The matching module is usually fully connected, it takes the two descriptors as input, and outputs a matching cost. Authors in [61] introduced classical

Siamese architecture for image patch matching. Later, others showed, that it is possible to replace the matching module by a non-parametric function such as L^2 [62] or cosine distance [19] and that the embedding modules may not share weights [63], and, finally, that the explicit notion of a descriptor might not be necessary [63].

Most of existing methods for training a Siamese network for patch matching are fully supervised, with the exception of [69] proposed in this work in Chapter 3. In most of cases, the training set consists of positive and negative examples. Each positive example (respectively negative) consist of a reference patch and its matching patch (respectively a non-matching one) from another image.

Training either takes one example at the time, positive or negative, and adapts the cost [61–64, 67], or takes at each step both a positive and a negative example, and maximizes the difference between the costs, aiming at making the cost for two patches from the positive example lower than the cost for two patches from the negative example [19, 65, 70]. We call this latter scheme as *Triplet Contrastive learning*.

Besides deep patch-matching cost, researcher used neural networks in other stereo reconstruction sub-tasks such as predicting a smoothness penalty in CRF model from a local intensity pattern [30, 71]. In [72] a network smooth the matching cost volume and predicts matching confidences, and in [73] the network detects and fixes incorrect disparities.

2.3.2 End-to-end approaches

Recent works attempt at solving stereo matching using a neural network trained end-to-end without post-processing [59, 74–80]. Such a network is typically a pipeline composed of *embedding, matching, regularization* and *refinement* modules.

The *embedding* module produces image descriptors for left and right images, and the (non-parametric) *matching* module performs an explicit correlation between shifted descriptors to compute a cost volume for every disparity [59, 74, 77–79]. Sometimes researchers omit this matching module and directly feed concatenated left-right descriptors to the *regularization* module [75, 76, 80]. This strategy uses more spatial context, but the deep network implementing such a module has a larger memory footprint.

The *regularization* module takes the cost volume, or the concatenation of the descriptors, regularizes it and outputs either disparities [59, 74, 77, 79] or distribution over disparities [75, 76, 78, 80]. In the latter case, researchers compute sub-pixel disparities as a weighted average with SoftArgmin, which is, however, sensitive to erroneous minor modes in the inferred distribution. The regularization module is usually a hourglass deep network with shortcut connections between the contracting and the expanding parts [59, 74–77, 79, 80]. In some cases [59, 74, 77, 79], it consists of 2d convolutions and does not treat all disparities symmetrically, which makes the network over-parametrized and prohibits the change of the disparity range without modification of its structure and re-training. In other cases, it consists of 3d convolutions that treat all disparities symmetrically [75, 76, 78, 80]. As a consequence, these networks have fewer parameters, but their disparity range is still non-adjustable without re-training due to the SoftArgmin.

Finally, some methods [77–79] also have a *refinement* module, that refines the initial low-resolution disparity relying on a attention map, computed as left-right warping error.

Most state-of-the-art methods rely on fully-supervised training on large synthetic datasets with ground truth [74], however, some works rely on *weakly supervised training* [76, 81, 82], that uses geometric constraints of the task. Most recent methods [82, 83] further improve results using *multi-task* learning.

All methods use modest-size image patches during the training and rely on L^1 loss, since it allows to train the network to produce sub-pixel disparities.

2.4 Weakly supervised methods

Although supervised training works well, the complexity of the deep models requires very large training sets with ground truth which are hard to collect in real applications. Besides, even when such large sets are available, the ground depth truth is often produced automatically from a depth sensor, such as LiDAR, and, thus, is usually noisy. The ground truth noise reduces the effectiveness of supervised training. One solution to this problem is to use synthetic training sets with the ground truth created using computer graphics [21, 74]. There is, however, significant *domain gap* between the synthetic training data and target domain data, that manifests in a large decrease in the performance when we deploy the network, pretrained on a synthetic set, in a *target domain* without fine-tuning. We can address this problem by using training methods with *weak supervision*, that we discuss in this section. First, in § 2.4.1 we describe different types of weak supervision, then in § 2.4.2 we review weakly supervised deep learning-based methods for the stereo matching.

2.4.1 Types of weak supervision

In this section, we focus on three types of methods with weak supervision: *weakly supervised*, *semi-supervised*, and *domain adaptation methods*.

In the *weakly supervised* methods, the ground truth is inexact or "coarse" and thus is easier or cheaper to collect. There are several approaches to weakly supervised learning. One approach is called *multi-instance learning* [84] and it uses labels for groups of training examples instead of labels for individual examples. Each group label indicates the presence of at least one example with a certain individual label in the group. This approach allows dealing with low geometrical accuracy or even the absence of geometrical information and labeling at the scene level [85]. Another interesting example of "coarse" supervisory data is a training set with unpaired input and ground truth examples [86]. These methods are called *cyclic consistency* *methods* and they work by forcing domain consistency of the predictions and consistent cyclic mapping from the input to the output domain and back.

Another approach is called *self-supervised learning* and it relies on the special structure of a problem that allows generating a ground truth automatically. This is, for example, possible in denoising [87], super-resolution, inpainting [88] and colorization [89] problems. Interestingly, these and similar problems [90–92] can be used for learning meaningful representations for others, unrelated tasks. In some cases, instead of automatically generating the ground truth it is possible to define losses, that do not require the ground truth at all. Such losses typically enforce constraints on the output such as, for example, temporal, spatial or semantic coherence [93].

In the *semi-supervised* methods the ground truth is incomplete, *i.e.* besides few examples with a ground truth, there is a large number of unlabeled examples. One approach to the semi-supervised learning is *self-training* [94, 95] and it operates by gradually including unlabeled examples with the high confidence labels, predicted by the method, to the training dataset. Another approach, is *transductive learning*, and it propagates the ground truth labels from the labeled to similar unlabeled training examples [96].

The *domain adaptation* [97–99] methods deal with the scenario when a model is trained in a *source domain*, with a sufficient amount of labeled training data, but deployed in a slightly different *target domain*, with a very few labeled examples or only unlabeled examples. This scenario takes place, for example, when one trains a deep model on a large synthetic dataset, such as [74, 100], and then deploys it in a real-world system. The main idea behind the transfer learning is to learn domain-invariant features that are equally useful in the source and in the target domains.

2.4.2 Deep weakly supervised stereo

The stereo matching problem has strong geometric constraints that can guide weakly supervised methods for training deep stereo networks. One of the first weakly supervised methods for deep stereo [69] is described in this dissertation in Chapter 3. It uses regularized predictions of the deep network as a ground truth. Among other early methods, self-training approach from [101] uses as ground truth, predictions of the deep method, satisfying one-to-one matching constraint, [102] employs as a ground truth most confident predictions of the non-learning based method [58], selected using the learning-based confidence measure [103].

Modern weakly supervised approaches in monocular [104–108] and stereo reconstruction [76, 81–83, 109, 110] mostly rely on a *image reconstruction*, *disparity consistency*, *disparity coherence* and *minimum disparity heuristic* self-supervised losses which we review below. These losses do not require ground truth disparity and can be defined for videos [111–113] or stereo pairs [76, 104–106, 113] and can also be used to train a network for predicting camera motion [111, 113, 114] and intrinsic parameters [112]. Interestingly, they even improve the results of the supervised training [82, 83].

The *image reconstruction or photometric* loss ensures that an image, warped to the opposite view using the estimated disparity, is similar to the actual opposite view image. The researchers warp the image using differentiable bi-linear interpolation, proposed in [115], and compute difference between the warped image and the actual image using pixel-wise L^2 [104, 116] or L^1 [81, 106–108, 112, 114] distances. Same objects often look different in stereo images because of differences in camera characteristics and parallax. Therefore, some methods use more robust image distances, such as *Structural Similarity Index (SSIM)* [76, 106, 107, 112, 114] or compare gradients of the images using L^1 distance [76] or warp images' features and compute the difference in the feature space [113]. Furthermore, the method from [81] uses GAN-based reconstruction loss that penalizes out-of-domain inconsistencies in the warped image and does not use the corresponding actual image.

The *disparity consistency* loss forces consistency of the disparities estimated for different views, *i.e.* enforces unique matching. This loss, requires disparities for two views. In [106], authors compute these disparities simultaneously and in [76, 107] in two separate runs. To compute the loss, in [81, 106], authors warp the estimated opposite view disparity and compare it to the estimated reference view disparity using L^1 distance. In [76], authors warp the reference view image twice using the estimated reference and the opposite view disparities and compare it to the original self.

The *disparity coherence* loss enforces spatial smoothness of the estimated disparity by penalizing L^2, L^1 norm or Charbonnier loss of first-order or second-order disparity gradients, sometimes attenuated in the locations with large first-order or second-order image gradients for edges preservation [76, 82, 104, 106, 107, 116]. In [83] authors compute the attenuation using the edge detection network and in [110] using dissimilarity of image and disparity patches centered at the corresponding locations. Furthermore, in [105, 108] the GAN-based loss penalizes out-of-domain inconsistencies in the estimated disparity.

The *disparity consistency* and the *image reconstruction* losses are invalid in the occluded areas. While most of the methods ignore this fact, in [82] authors implicitly handle the occlusions by ignoring large pixel-wise reconstruction losses, and in [112, 117] explicitly detect and ignore the occlusions.

The *minimum disparity heuristic* loss encourages the network to prefer smaller disparity in the occluded areas by penalizing L^1 norm of the estimated disparity [76, 107].

Recently, several novel losses were proposed. In [110] the authors proposed a *scale consistency* loss, that encourages the network to predict the same disparity for up-scaled and down-scaled versions of the same stereo images, in [82] the authors proposed a *semantic consistency loss* that forces consistent segmentation in different stereo views.

Several *domain adaptation* methods train a network in a source domain and deploy it in a

Table 2.1 – Summary of stereo datasets that we use in this work.

Dataset	Test #	Train #	Size	Max disp.	Ground truth	Web score
Middlebury	60	15	1.5-6 MP	30-800	dense, ≤ 0.2 pix.	✓
KITTI	395	395	1226 × 370	192	sparse, ≤ 3 pix.	✓
FlyingThings3D	4370	25756	960 × 540	192	dense	✗

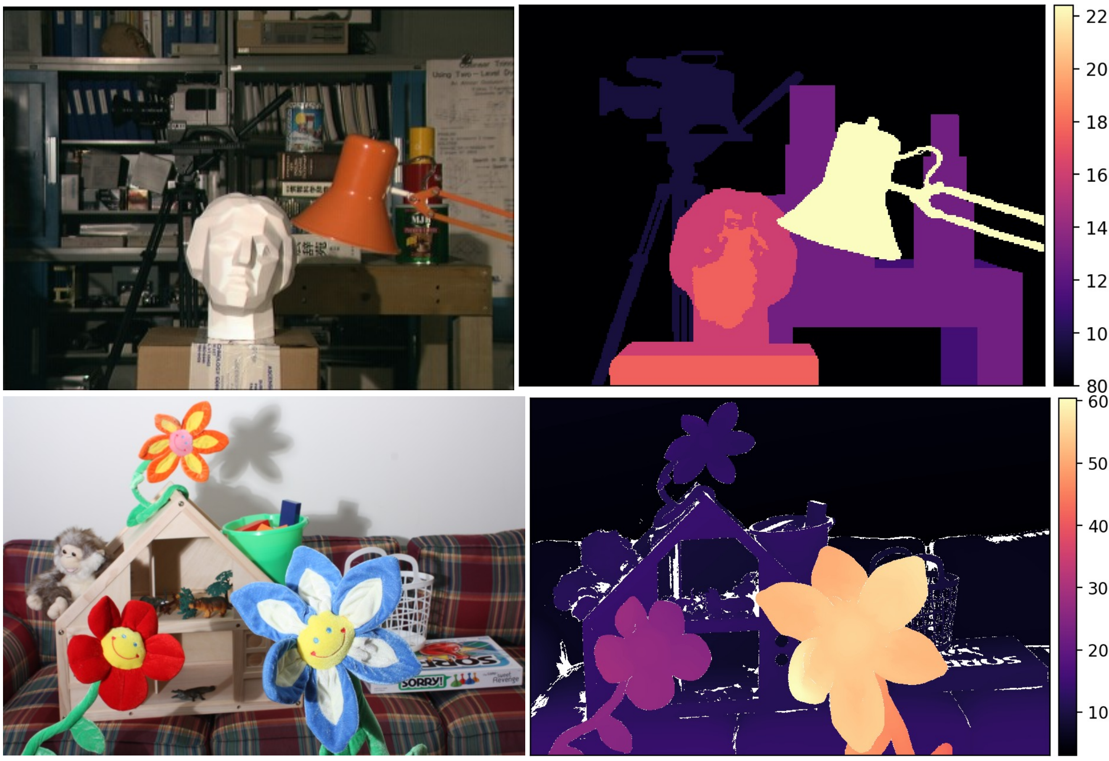
slightly different target domain. In [108] the authors apply *style transfer* to target domain images to make them more similar to the synthetic, source domain images, in [105, 108] the authors use *domain confusion loss* from [118], to prevent over-fitting to the source domain. The main drawback of these approaches is that they do not allow to discover depth cues, specific to a target domain.

2.5 Stereo datasets

In this section, we provide detailed information about stereo datasets that we use in this work. There are three popular benchmark datasets: KITTI [5, 119], Middlebury (MB) [29, 120–122] and FlyingThings3d [74], summarized in Table 2.1. KITTI and Middlebury datasets have online scoreboards [123, 124], showing comparative performance of all participating stereo methods. KITTI dataset additionally has unlabeled training data, which we use in our experiments with weakly supervised learning.

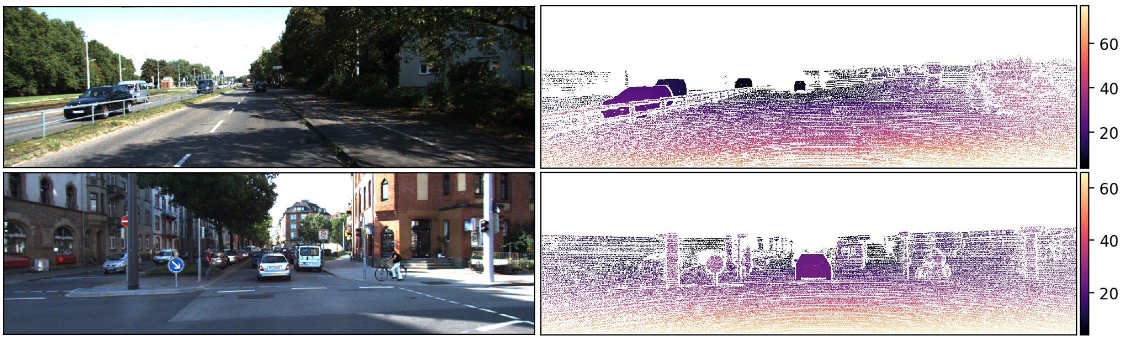
Middlebury (MB) is the oldest benchmark datasets in the field. Researchers mostly use it for evaluation of non-learning based stereo methods since it does not have enough labeled data for training deep neural networks. We show typical examples from the dataset in Figure 2.6. The dataset consists of five subsets published in separate works [29, 120–122]. In total, it has 60 training and 15 test scenes. Each subsets consist stereo images of staged scenes acquired by different stereo systems. For some scenes, there are several stereo pairs acquired under different exposures and illuminations. The size of images varies from 1.5 to 6 Mega Pixels (MP) and the disparity range varies from 30 to 800 pixels depending on the scene. In this work, we use half-resolution images due to GPU size limitation. The training examples include a dense ground truth disparity for the left and right views acquired by a structured light system with error < 0.2 pixels. The dataset has online scoreboard [124], where methods are ranked according to their performance on the test set. To participate in the evaluation, developers upload their test set results to the scoreboard. Only a single upload per method is allowed. After the upload, the web site computes test performance as 2-pixels-error rate in non-occluded locations using ground truth disparity, hidden from the participants.

KITTI is a larger and more realistic dataset. We show typical examples from the dataset in Figure 2.7. The dataset consists of two subsets published in the separate works: KITTI 2012 [119] and KITTI 2015 [5]. The KITTI 2012 subset consists of 200 training and test examples and the KITTI 2015 subset consists of 195 training and test examples. Every examples is a rectified color stereo pairs with resolution 1226×370 (approximately) acquired from cars moving around a



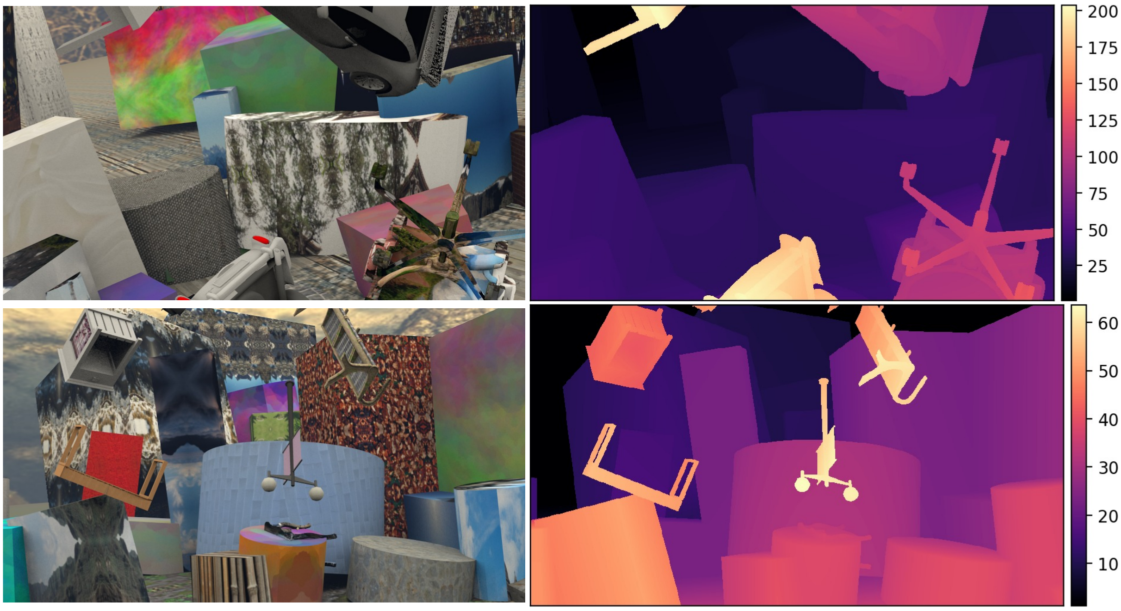
(a) Left image (b) Left view disparity

Figure 2.6 – Typical examples from the Middlebury dataset. In the disparity images, the white pixels correspond to the locations without the ground truth and the warmer colors correspond to larger disparities.



(a) Left image (b) Left view disparity

Figure 2.7 – Typical examples from the KITTI dataset. In the disparity images, the white pixels correspond to the locations without the ground truth and the warmer colors correspond to larger disparities. The reflective surfaces, such as car glasses, are densely labeled using car models.



(a) Left image (b) Left view disparity

Figure 2.8 – Typical examples from the FlyingThings3d dataset. The warmer colors in the disparity maps correspond to larger disparities.

city. About 30% of the pixels in the training set have a ground truth disparity acquired by a LiDAR with an error < 3 pixels. The disparity range is about 230 pixels. For reflective surfaces, such as car glasses, the dataset provides dense labels, based on car models fitting. Note, that in the older KITTI 2012 subset the ground truth disparities for reflective surfaces are located in a separate folder. The KITTI 2015 subset provides ground truth disparities for the left and the right views, while KITTI 2012 provides ground truth only for the left view. Both subsets have online scoreboards [123], where all methods are ranked according to their performance on the test set. The scoreboard computes the test performance using ground truth disparity, hidden from the participants using 3-pixels-error rate in non-occluded locations for KITTI 2012, and all locations for KITTI 2015.

Each subset of KITTI dataset has an extension (respectively KITTI-EXT 2012 and KITTI-EXT 2015) that contains 19 additional stereo pairs for each scene, without ground truth disparity, *i.e.* 20× more unlabeled training data. We use this data in our experiment with weakly supervised learning in Chapter 3.

FlyingThings3d [74] is a very large synthetic dataset produced using computer graphics. We show typical examples from the dataset in Figure 2.8. Each scene in the dataset depicts various things flying in the air, overlapping and occluding each other. The dataset contains 25K training and 4k test pairs, each supplied with precise ground truth disparities for the left and the right views. The dataset does not have an online scoreboard.

The FlyingThings3d set suffers from two problems: (1) as noticed in [76, 77], some images have very large (up to 10^3) or negative disparities and (2) some images are rendered with artifacts. To deal with these problems, in some previous publications authors process the test set using the ground truth. For example, authors in [80] ignore pixels with disparity > 192, while authors in [74, 77, 79] discard images where more than 25% of pixels have disparity > 300. In this work, during the training, we use only images without artifacts and with the disparities $\in [0, 255]$ and during the benchmarking we use two evaluation protocols mentioned above for fairness of the comparison.

2.6 Accuracy measures

In this work, we measure accuracy of stereo reconstruction using three standard measures:

- *n-pixels-error* (NPE), which is a percentage of pixels for which the predicted disparity is off by more than *n* pixels. We mostly use *3-pixels-error* (3PE) and *1-pixel-error* (1PE),
- *mean absolute error* (MAE) is an average absolute difference between the predicted and the ground truth disparities, and
- *mean depth error* (MDE) is an average absolute difference between the predicted and the ground truth depth.

Note, that the NPE and the MAE are complimentary, since the NPE characterize error robust to outliers, while MAE accounts for sub-pixel error.

Research part Part I

3 Weakly supervised learning of deep patch-matching cost

Contents

3.1 Introduction **28**

3.2 Method **28**

3.2.1 Problem formulation 28

3.2.2 Stereo matching constraints 29

3.2.3 Weakly supervised methods 30

3.3 Experiments **34**

3.3.1 Parameters tuning 35

3.3.2 Comparison of the proposed methods 36

3.3.3 Comparison with the supervised method 36

3.3.4 Benchmarking 37

3.3.5 What does the deep matching cost network learn? 37

3.3.6 Weakly supervised domain adaptation 39

3.3.7 Training with a noisy ground truth 40

3.4 Conclusion **41**

This chapter is based on the following publication:

S. Tulyakov, A. Ivanov, and F. Fleuret. Weakly supervised learning of deep metrics for stereo reconstruction. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1339-1348. 2017.

3.1 Introduction

Deep learning-based costs have demonstrated extremely good performance for matching image patches for stereo reconstruction. However, training networks for computing such costs requires a large amount of training data with ground truth disparity, which can be difficult or costly to collect for certain applications, such as, for example, satellite stereo imaging. In this work, we address this problem by introducing a new weakly supervised method that does not require the ground truth disparity but, instead, relies on coarse information about scenes and a stereo system.

Main contributions of this work are following:

1. We introduce a new weakly supervised method for learning deep patch-matching cost. Our method alternatively computes matches using current network for matching cost computation, regularizes and uses them as ground truth to improve the network. The method allows learning a high-quality deep patch-matching cost when labeled training data is not available or contaminated with noise.
2. We experimentally show that for given network architecture, training with this new method without ground truth produces a cost with performance as good as state-of-the-art baselines trained with the said ground truth across several reference datasets.

This chapter is organized as followings: First, in the “Method” section we describe the proposed weakly supervised method in details: in § 3.2.1 we formulate the task of weakly supervised learning of patch-matching cost for stereo, in § 3.2.2 review stereo matching constraints, and in § 3.2.3 propose several weakly supervised learning methods that use these constraints.

Next, in the “Experiments” section we show effectiveness of the proposed approach: in § 3.3.1 we choose meta-parameters of the proposed methods, in § 3.3.2 compare the proposed weakly supervised methods to each other, in § 3.3.3 compare the best-performing weakly supervised method to the supervised baseline method, in § 3.3.4 benchmark stereo matching method with the proposed patch-matching cost against state-of-the-art stereo methods, in § 3.3.5 visualize what the deep network learns when trained with the proposed method, in § 3.3.6 show ability of the proposed method to adapt a deep patch-matching cost to domains without the ground truth, and in § 3.3.7 demonstrate advantages of the proposed method in presence of a ground truth noise. Finally, in the “Conclusion” section we summarize our results.

3.2 Method

3.2.1 Problem formulation

We are provided with a weakly supervised training set, where each training example is a triplet (P^L, P^R, P^Ξ) of series of $s \times s$ gray-scale patches:

- $P^L = (p_1^L, p_2^L \dots p_w^L)$ is a series of patches extracted from a horizontal line of left rectified stereo image,
- $P^R = (p_1^R, p_2^R \dots p_w^R)$ is a series of patches extracted from the same horizontal line of the corresponding right rectified stereo image, and
- $P^\Xi = (p_1^\Xi, p_2^\Xi \dots p_w^\Xi)$ is a series of patches, extracted from a random, non-matching horizontal line of the corresponding right rectified stereo image,

where w is a number of patches per line, or, equivalently, a width of a training image. In addition to the training set, we are provided with a maximum possible disparity d_{max}, which depends on a stereo system and prior knowledge about a scene.

For a series of left P^L and right P^R image patches, a *deep matching cost* network with parameters $\mathbf{\Theta}$ computes a *matching cost matrix* $\mathbf{c}^{LR}$ of size $w \times w$ as

$$\mathbf{c}^{LR} = \mathrm{Net}(P^L, P^R \mid \mathbf{\Theta}, d_{max}) \text{ , where } c_{ji}^{LR} = \begin{cases} \mathrm{Net}(p_j^L, p_i^R \mid \mathbf{\Theta}) & 0 \le j - i \le d_{max} \\ -\infty & \text{otherwise} \end{cases} \tag{3.1}$$

Given the matching cost matrix it is possible to estimate a *disparity vector* $\widehat{\mathbf{d}}$ of size w as

$$\widehat{d_j} = \arg\min_d c_{j,j-d}^{LR} \tag{3.2}$$

The goal of this work is to find parameters $\mathbf{\Theta}$ of the network such that the estimated disparity vector $\widehat{\mathbf{d}}$ is equivalent to the corresponding ground truth disparity vector $\mathbf{d}^{GT}$, which is not available during the training.

Note, that unlike [21, 61–64, 66, 67] which utilize pairs of patches for the training, in this work we use triplets of series of patches. This allows using constraints defined on a group of patches. Additionally, processing the series of patches as a whole significantly speeds up the training process by allowing to reuse shared computations.

3.2.2 Stereo matching constraints

The stereo matching problem has very strong geometric constraints that can be used for the weakly supervised learning. In this work we use the following constraints:

(E) *Epipolar constraint.* Every non-occluded left image patch has a matching right image patch on the corresponding epipolar line [22][239-241p].

(D) *Disparity range constraint.* Horizontal offset of a left patch relative to the matching right patch is bounded by a maximum disparity d_{max}, which depends on the parameters of a stereo system (focal length, pixel size, baseline) and distance range of the scenes.

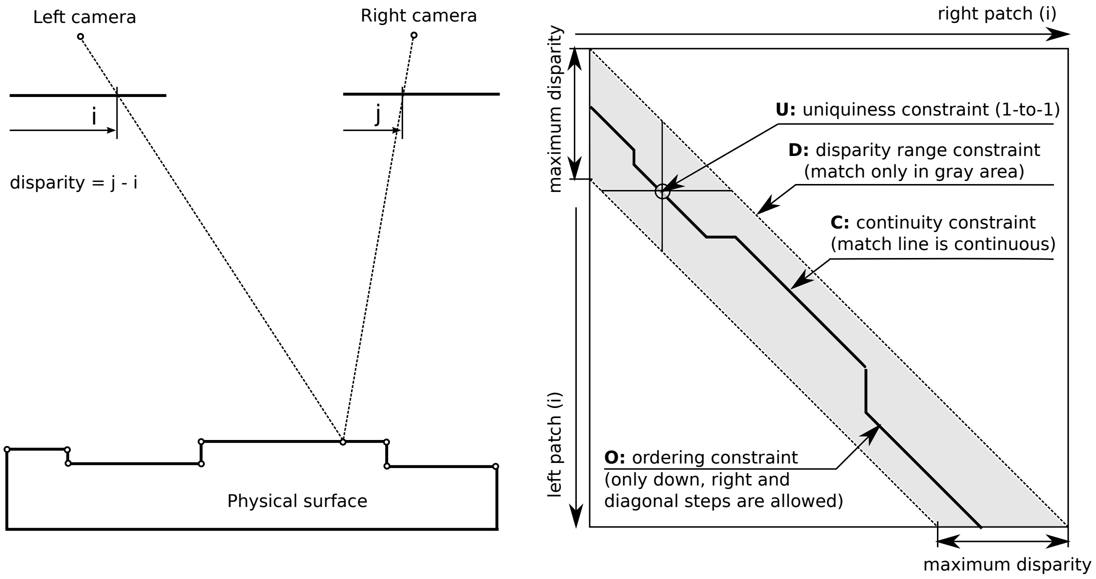

Left camera
Right camera
i
j
disparity = j - i
Physical surface
right patch (i)
maximum disparity
left patch (i)
U: uniquiness constraint (1-to-1)
D: disparity range constraint
(match only in gray area)
C: continuity constraint
(match line is continuous)
O: ordering constraint
(only down, right and
diagonal steps are allowed)
maximum disparity

Figure 3.1 – The left figure shows a scene and the right figure shows the corresponding matching cost matrix. The bold line corresponds to the true matches that satisfy the stereo constraints. Elements within the disparity range are shown in gray. Note that for some points there are no matches.

(U) *Uniqueness constraint.* Matching pair of patches is unique [125].

(C) *Continuity (smoothness) constraint.* The horizontal offsets of left image patches relative to the matching right image patches are similar for nearby patches everywhere except on depth discontinuities [125].

(O) *Ordering constraint.* Order of left image patches on the epipolar line is similar to the order of the matching right image patches on their epipolar line.

These constraints result in a particular shape of a matching cost matrix shown in Figure 3.1. Note that the uniqueness (U), the continuity (C) and the ordering (O) constraints are sometimes violated. However, our experiments show that these rare violations only marginally affect the training in the presence of a large training set.

3.2.3 Weakly supervised methods

We developed several weakly supervised methods that use different subsets of the stereo constraints. They can be used in combination with any network architecture for patch-matching cost computation and any gradient-based optimization method.

All proposed methods alternate between two steps

(1) Compute unconstrained p^{R-} and constrained p^{R+} matching patches, subject to constraints $\mathscr{C}$, for the left image patch p^L, given current parameters of the patch-matching

network

$$
\begin{aligned}
p^{R-} &= \operatorname*{arg\,min}_{p^R \in P^R} \operatorname{Net}(p^L, p^R \mid \boldsymbol{\Theta}) \\
p^{R+} &= \operatorname*{arg\,min}_{p^R \in P^R, \mathscr{C}(p^R)} \operatorname{Net}(p^L, p^R \mid \boldsymbol{\Theta})
\end{aligned}
\tag{3.3}
$$

(2) Update network parameters $\boldsymbol{\Theta}$ to make the network produce lower matching cost for the constrained match, than for the unconstrained match by minimizing *triplet contrastive loss*, as

$$
\begin{aligned}
\widehat{\boldsymbol{\Theta}} &= \operatorname*{arg\,min}_{\boldsymbol{\Theta}} \operatorname{Loss}(\boldsymbol{\Theta} \mid p^L, p^{R+}, p^{R-}) \\
\operatorname{Loss}(\boldsymbol{\Theta}) &= \max(0, \operatorname{Net}(p^L, p^{R+} \mid \boldsymbol{\Theta}) - \operatorname{Net}(p^L, p^{R-} \mid \boldsymbol{\Theta}) + \lambda)
\end{aligned}
\tag{3.4}
$$

In most of the proposed methods, the first step is “integrated” in the loss function. As a result the loss function takes as an input one or several following matrices: $\mathbf{c}^{LR} = \operatorname{Net}(P^L, P^R \mid \boldsymbol{\Theta}, d_{max})$, $\mathbf{c}^{L\Xi} = \operatorname{Net}(P^L, P^\Xi \mid \boldsymbol{\Theta}, d_{max})$ and $\mathbf{c}^{R\Xi} = \operatorname{Net}(P^R, P^\Xi \mid \boldsymbol{\Theta}, d_{max})$.

Multi-Instance Learning (MIL). This method is inspired by Multi-Instance Learning paradigm explained in § 2.4.1 and relies only the epipolar (E) and the disparity range constraints (D) from § 3.2.2.

From these two constraints, we know that every non-occluded left image patch has a matching patch on the corresponding right epipolar line in a known index interval, but does not have a matching patch on a random epipolar line. Therefore, for every left image patch, the lowest matching cost for a patch on the corresponding right epipolar line should be lower than the lowest matching cost for the patch on a random epipolar line. Our training objective to push apart these two costs as

$$
\begin{aligned}
\operatorname{Loss}(\boldsymbol{\Theta}) = &\frac{1}{|\text{rows}|} \sum_{j \in \text{rows}} \max(0, \min_i c_{ji}^{LR} - \min_i c_{ji}^{L\Xi} + \lambda) + \\
&\frac{1}{|\text{cols}|} \sum_{i \in \text{cols}} \max(0, \min_j c_{ji}^{LR} - \min_j c_{ji}^{L\Xi} + \lambda),
\end{aligned}
\tag{3.5}
$$

where rows $= [d_{max} \ldots w)$ and cols $= [0 \ldots w - d_{max})$ are sets of rows and columns of the matching cost matrices that are guaranteed to have correct matches (see Figure 3.1), and λ is a *hinge loss margin*, which we set equal to 0.2, based on a small-scale grid search. Note that the disparity range constraint is taken into account automatically when we compute the cost matrices using Equation 3.1.

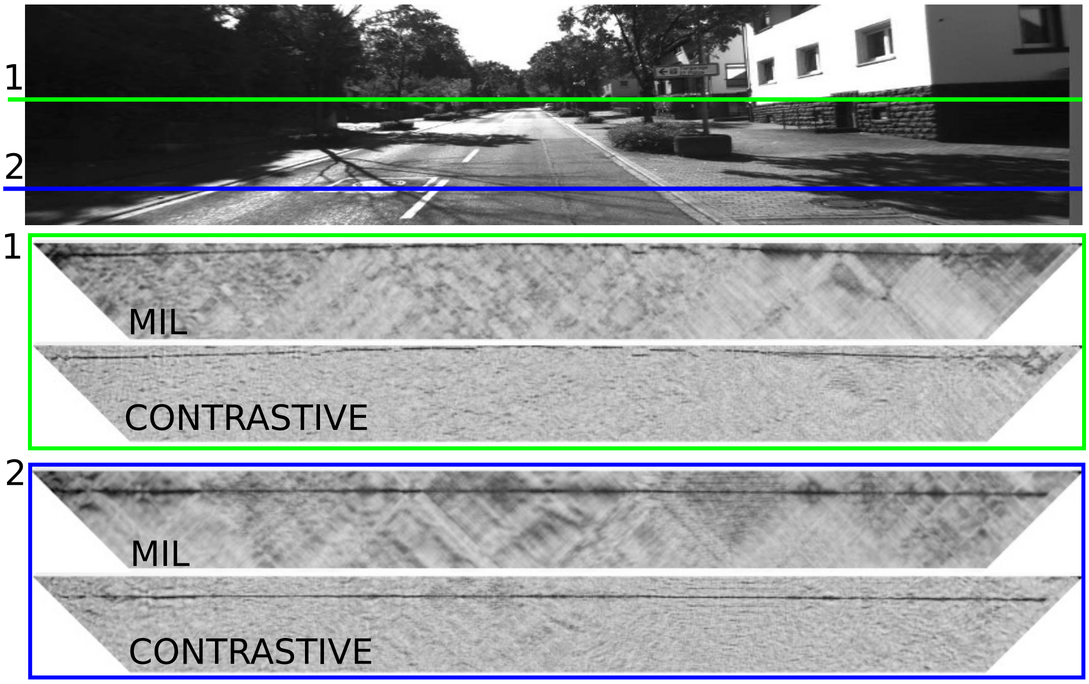


Figure 3.2 – The problem of Multi-Instance Learning (MIL) method. The top figure shows a left stereo image from the KITTI 2012 dataset with two highlighted epipolar lines. The pictures below show the diagonal part of the matching cost matrices obtained with the MIL and Contrastive methods for these epipolar lines. The dark elements in the cost matrices correspond to the smaller cost. WTA error for the MIL method is 18.45%, and for the Contrastive method is 17.63%. Note that in the case of the MIL method, the cost matrices are very blocky. The blocks correspond to the areas where the matching cost does not allow identification of the unique matches. The problem is solved in the Contrastive method.

According to our experiments, the main problem of the method is that it learns matching cost insensitive to small shifts from the optimal match. This problem manifests in a blocky shape of a cost matrix, where the blocks correspond to the areas where the cost does not allow identification of a unique match as shown in Figure 3.2. This issue motivates the Contrastive method described in the following section that solves the problem by explicitly using the uniqueness constraint (U) during the training.

Contrastive method uses the epipolar (E), the disparity range (D), and the uniqueness constraints (U) from § 3.2.2.

From the epipolar and the disparity range constraints we know that every non-occluded left image patch has a matching patch on the corresponding right epipolar line in a known index interval. Furthermore, according to the uniqueness constraint, the matching patch is unique. Therefore, for every left image patch, the lowest matching cost for a patch on the corresponding right epipolar line should be much lower than the second-lowest matching cost. Our training objective is to push apart these two quantities as

$$
\begin{aligned}
\text{Loss}(\boldsymbol{\Theta}) = & \frac{1}{|\text{rows}|} \sum_{j \in \text{rows}} \max(0, \min_i c_{ji}^{LR} - \min_i \widehat{c}_{ji}^{LR} + \lambda) + \\
& \frac{1}{|\text{cols}|} \sum_{i \in \text{cols}} \max(0, \min_j c_{ji}^{LR} - \min_j \check{c}_{ji}^{LR} + \lambda), \qquad (3.6)
\end{aligned}
$$

where $\widehat{\mathbf{c}}^{LR}$ and $\check{\mathbf{c}}^{LR}$ are matching cost matrices with masked out with $-\infty$ row-wise and column-wise minima respectively.

Training with the Contrastive method presented above requires good initialization of the cost network. Fortunately, in this work we use the network which works very well with random CNN weights, as shown in § 3.3.5, thus we can use the Contrastive method right from the start. In other cases, it would be necessary to pre-train a network with another method such as MIL.

According to our experiments, the Contrastive method suffers from a problem opposite to the problem of the MIL method: it produces over-sharpened cost, sensitive even to small shifts from the exact match. This is also detrimental to the performance since our goal is to find cost invariant to small geometric transformations, such as shift. We solve the problem by masking out all spatial neighbors within a *non-maximum suppression radius* r_{sup} from minima in $\widehat{\mathbf{c}}^{LR}$ and $\check{\mathbf{c}}^{LR}$ with $-\infty$. See the parameters tuning experiments in § 3.3.1 for details.

Contrastive-DP method uses all constraints listed in § 3.2.2. Its main difference with the Contrastive method is that this method contrasts unconstrained match to the match under the continuity (C) and the ordering (O) constraints computed using *Dynamic Programming (DP)*, instead of the best and the second-best matches.

Formally, the DP finds the global matching path $\widehat{M} = ((j_1, i_1), (j_2, i_2), \ldots)$ which is a series of pairwise patch matches (bold line in Figure 3.1) by solving the following optimization problem

$$
\widehat{M} = \underset{M \in \mathbb{M}}{\arg\min} \frac{1}{|M|} \sum_{(j,i) \in M} c_{ji}^{LR}, \qquad (3.7)
$$

where $|M|$ is a number of pairwise patch matches in the global match, $\mathbb{M}$ is a set of all global matches which are continuous in the following sense

$$
\begin{aligned}
& \forall k > 1, (j_{k+1}, i_{k+1}) - (j_k, i_k) \in \{(0,1), (1,0), (1,1)\}, \\
& \text{and } \ (j_1, i_1) \in \{1\} \times [0, d_{max}]. \qquad (3.8)
\end{aligned}
$$

This means that only down, right, and diagonal steps are allowed, which enforces the continuity (C) and the ordering (O) constraints in the solution. Notice also that we search for a path that has a minimum average cost rather than a minimum total cost to prevent bias toward global matches consisting of more pairwise matches.

Given the best global match $\widehat{M}$ found by the DP, we define the loss function as

$$\begin{aligned}\mathrm{Loss}(\mathbf{\Theta}) = & \frac{1}{|\widehat{M}|}\sum_{(j,i)\in\widehat{M}} \min(0, c^{LR}_{ji} - \max_k \widetilde{c}^{LR}_{jk} + \lambda) + \\ & \frac{1}{|\widehat{M}|}\sum_{(j,i)\in\widehat{M}} \max(0, c^{LR}_{ji} - \min_k \widetilde{c}^{LR}_{ki} + \lambda),\end{aligned} \tag{3.9}$$

where $\widetilde{\mathbf{c}}^{LR}$ is a cost matrix where all neighbors of elements belonging to the optimal global match $\widehat{M}$ within a *non-maximum suppression radius* r_{sup} are masked with $-\infty$.

The optimal global match computed by the DP might contain vertical and horizontal segments, that correspond to depth discontinuities, violating the uniqueness constraint (U). During the training, we ignore all such segments that are longer than a *maximum occlusion segment length* ℓ_{occ}. See the parameters tuning experiments in § 3.3.1 for details.

The proposed method uses a DP to regularize a noisy prediction of a deep matching cost network as it is currently trained. The similar idea in a different context appeared in [93]. In both [93] and our method, the high-level idea is to learn an embedding in a weakly supervised manner by minimizing the energy of the best path on a constraint graph, while simultaneously maximizing the energy of the best unconstrained path. However there are important differences: (1) we use per location loss function with a margin, while they use per path loss function without the margin, (2) we use stereo constraints to construct the constrained graph, while they use text string labels, and finally (3) the application domains are drastically different since we deal with geometrical regression and they deal with classification.

3.3 Experiments

We guarantee reproducibility of all experiments in this section by using only available datasets, and making our code available online[1].

In all experiments in this work, we use Torch7 framework [126, 127] and adopt architecture of deep patch-matching cost network from [66], shown in Figure 3.3. We perform initialization of weights and biases of the network in a standard way by random sampling from a zero-mean uniform distribution. For the optimization, we use ADAM method with standard settings and mini-batches of size equal to the training image height, and no data augmentation. In the experiments, we use KITTI and Middlebury datasets described in Chapter 2.

We measure an accuracy of the deep matching cost in two scenarios. In the first scenario, which we call *winner-takes-all (WTA)*, to compute the disparity we simply pick the disparity with the lowest matching cost. In the second scenario, which we call *pipeline*, we plug-in the estimated matching cost into the legacy stereo matching pipeline from [19], which in turn produces disparity. Note that we do not explicitly optimize the matching cost for performance

---

[1] https://github.com/tlkvstepan/mc-cnn-ws

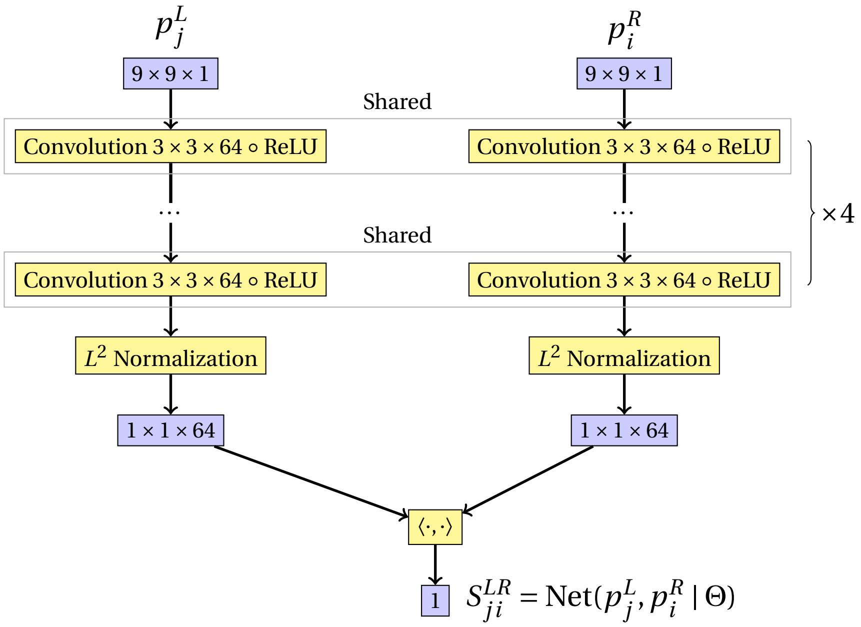

p_j^L
p_i^R
9 × 9 × 1
9 × 9 × 1
Shared
Convolution 3 × 3 × 64 ∘ ReLU
Convolution 3 × 3 × 64 ∘ ReLU
Shared
Convolution 3 × 3 × 64 ∘ ReLU
Convolution 3 × 3 × 64 ∘ ReLU
×4
L^2 Normalization
L^2 Normalization
1 × 1 × 64
1 × 1 × 64
⟨·,·⟩
1
$S_{ji}^{LR} = \mathrm{Net}(p_j^L, p_i^R \mid \Theta)$

Figure 3.3 – The architectures of the deep matching cost network from [66] that we use in our experiments.

Table 3.1 – Parameters tuning for the Contrastive-DP method on the KITTI 2012 validation set. The parameters with the smallest error are shown in bold.

Parameter	**Values**	**Winner-take-all 3PE, [%]**
Non-max suppression radius r_{sup}	0, 1, **2**, 4, 8, 16	21.81, 16.83, **16.66**, 16.86, 17.02, 17.28
Max. occlusion segment ℓ_{occ}	**1**, 2, 4, 8, 16	**16.93**, 17.32, 17.62, 17.58, 17.56

within the pipeline. The disparity accuracy is measured using 3-pixels-error rate, described in Chapter 2.

3.3.1 Parameters tuning

Our methods rely on two meta-parameters: the *non-maximum suppression radius* r_{sup} and the *maximum occlusion segment length* ℓ_{occ}. To select the optimal value for each, we keep all but one parameter fixed and run the training procedure for a small number of iterations on the KITTI 2012 dataset. Then we compare the WTA errors on the validation set.

The results, shown in Table 3.1, confirm our guess from § 3.2.3 that it is better to mask elements in cost matrix within the radius $r_{sup} = 2$ from the optimal match. They also confirm that it is better to ignore occluded segments larger than $\ell_{occ} = 1$.

Table 3.2 – Comparison of the proposed weakly supervised learning methods on the KITTI 2012 validation set. All methods are used to train the same network architecture. We list constraints used by each method, using capital letters, where E indicates epipolar, D - disparity range, U - uniqueness, O - ordering, and C - continuity constraint. The highlighted Contrastive-DP method uses all stereo matching constraints during learning and achieves the smallest WTA 3PE. Notice that in general, methods that use more constraints perform better.

Method	**Winner-take-all 3PE,** [%]	**Time, [hr]**
Multi-Instance Learning (E, D)	18.45	45
Contrastive (E, D, U)	17.63	30
Contrastive-DP (E, D, U, O, C)	14.61	68

Table 3.3 – Comparison of the proposed weakly supervised method with the fully-supervised baseline [66] on the validation sets. In the table, we highlight the method with the smallest WTA errors. The proposed method outperforms the baseline in terms of WTA error across all datasets. This is remarkable since, in contrast to the supervised method, it does not use ground truth disparity during learning. For reference, the two bottom rows show the performance of two standard hand-crafted matching costs, where SAD stands for the sum of absolute differences of pixel intensities in 9 × 9 image patch. Note that following the setup of [66], as input to the deep-learning methods we use patches of size 9 × 9 for the KITTI 2012 and 2015, and 11 × 11 for the MB.

Method	**Winner-take-all 3PE, [%]**		
	KITTI 2012	**KITTI 2015**	**Middlebury**
MC-CNN fst [66]	15.44	15.38	29.94
MC-CNN-WS fst (proposed)	13.90	14.08	29.60
CENSUS 9 × 9 [47]	53.52	50.35	64.53
SAD 9 × 9	32.36	30.67	59.39

3.3.2 Comparison of the proposed methods

In this experiment, we compare the performance of the proposed weakly supervised methods on the KITTI 2012 dataset using the winner-take-all (WTA) 3PE. The results of the experiment are shown in Table 3.2.

The main conclusion is that weakly supervised methods that use more stereo constraints for the training perform better. For example, the MIL, that uses only the epipolar and the disparity range constraints, has larges WTA 3PE, whereas the Contrastive-DP, that uses all constraints has the smallest error. In all following sections, we use the best performing Contrastive-DP method only and refer to it as MC-CNN-WS, where WS stands for weakly supervised.

3.3.3 Comparison with the supervised method

In this section, we compare the proposed weakly supervised method with the fully supervised baseline [66] on the three different sets, using the winner-take-all (WTA) 3-pixels-error (3PE) (see § 2.6 and § 3.3).

The results are shown in Table 3.3. As we see, the proposed method outperforms the fully-supervised method in terms of WTA error across all three sets. This is remarkable because the method does not use ground truth disparity during learning. The success of our method in case of the KITTI 2012 and the KITTI 2015 sets can be attributed to the fact that these sets have a large amount of unlabeled stereo data, that can be used by our method. These sets have more than 20× more unlabeled data than labeled training data. In the case of MB, dataset our method does not have such a huge advantage over the supervised method. The set has only 30% more unlabeled training data than the labeled training data.

3.3.4 Benchmarking

In this section, we investigate how well our weakly supervised deep matching cost network performs when we plug it into the stereo pipeline from [66]. First, we tune the parameters of the pipeline for each dataset using a simple coordinate descent method, starting from the default values of [66]. Then we compute disparity maps for the test sets with withheld ground truth and upload the results to the evaluation web sites [123, 124]. The evaluation results are available in online scoreboards and shown in Tables 3.5 and 3.4 (note, that corresponding disparity maps are also available for viewing in the scoreboards).

As we can see, across all benchmarks the results with matching cost network trained without ground truth are very close to the results with matching cost network in the fully-supervised manner. Those are very encouraging results, given in particular that we do not optimize the matching cost network and the pipeline parameters together, and considering the performance in the winner-take-all setup of § 3.3.3. The fact that our cost outperforms the supervised cost in winner-takes-all setup but lags behind it when used as a part of the pipeline is not surprising. The pipeline relies on multiple heuristics and thus provide a regularization that is not taken into account during the training of the cost network. Thus, smaller WTA error does not guarantee smaller Pipeline error.

Regarding the inference time, note that since we use the same pipeline and the network as [66], the inference times are also very similar. The differences are only due to the hardware differences.

3.3.5 What does the deep matching cost network learn?

In Figure 3.4, we show matching cost matrices computed before and after the training with the proposed MC-CNN-WS method on the KITTI 2012 dataset. While one can not visually distinguish the best match in the cost matrices before the training, it becomes visible after the training. This suggests that the training improves the discriminative ability of the deep matching cost. Notably, the performance of the matching cost network with random weights is surprisingly good. The corresponding WTA error on the KITTI 2012 is just 42.01%. This good

Table 3.4 – Middlebury benchmark [124] snapshot from 14/11/2016 with published methods (default view). The proposed MC-CNN-WS method, highlighted in the table, does not use ground truth data during the training but has the error rate very similar to the error rate of the supervised MC-CNN fst method, shown in bold, trained with the ground truth data.

Rank	Submission	Method	2PE (non-occluded), [%]	Time, [s]
1	19/01/2015	NTDE [128]	7.62	300
2	28/08/2015	MC-CNN acrt [19]	8.29	254
3	03/11/2015	MC-CNN+RBS [129]	8.62	345
4	**26/01/2016**	**MC-CNN fst [19]**	**9.69**	**2.94**
5	11/14/2016	MC-CNN-WS (proposed)	12.1	5.59
6	13/10/2015	MDP [130]	12.6	130
7	19/04/2015	MeshStereo [131]	13.4	146

Table 3.5 – KITTI 2012 (top) and KITTI 2015 (bottom) benchmark [123] snapshots from 14/11/2016 with published methods (default view). The proposed MC-CNN-WS method, highlighted in the tables, does not use ground truth data during the training but has the error rate very similar to that of the supervised MC-CNN fst method, shown in bold, trained with the ground truth. Since the MC-CNN fst method does not appear on the evaluation tables, due to restrictions on the number of results for a single paper, we borrow the results from [19]

Rank	Submission	Method	3PE (non-occluded), [%]	Time, [s]
1	27/04/2016	PBCP [132]	2.36	68
2	26/10/2015	Displets v2 [40]	2.37	265
3	21/08/2015	MC-CNN acrt [19]	2.43	67
4	30/03/2016	cfusion [133]	2.46	70
5	16/04/2015	PRSM [134]	2.78	300
6	**21/08/2015**	**MC-CNN fst [19]**	**2.82**	**0.8**
7	03/08/2015	SPS-st [41]	2.83	2
8	14/11/2016	MC-CNN-WS (proposed)	3.02	1.35
9	03/03/2014	VC-SF [135]	3.05	300

Rank	Submission	Method	3PE (all pixels), [%]	Time, [s]
1	26/10/2015	Displets v2 [40]	3.43	265
2	27/04/2016	PBCP [132]	3.61	68
3	21/08/2015	MC-CNN acrt [19]	3.89	2.94
4	16/04/2015	PRSM [134]	4.27	300
5	06/11/2015	DispNetC [74]	4.34	0.06
6	11/04/2016	ContentCNN [136]	4.54	1
7	**21/08/2015**	**MC-CNN fst [19]**	**4.62**	**0.8**
8	14/11/2016	MC-CNN-WS (proposed)	4.97	1.35
9	03/08/2015	SPS-st [41]	5.31	2

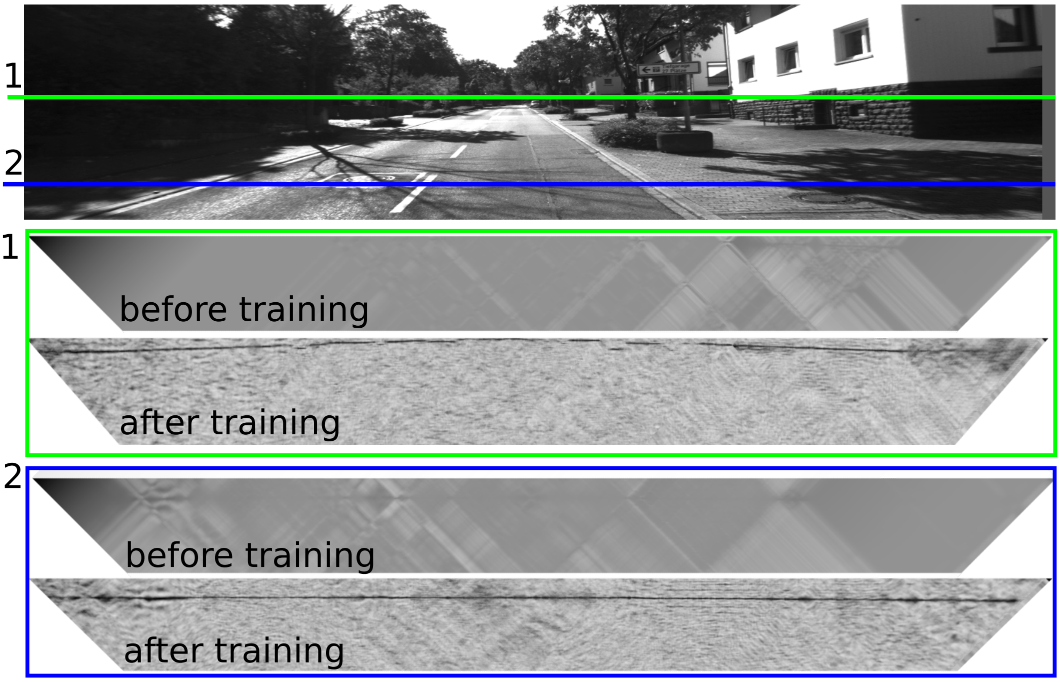


Figure 3.4 – Diagonal part of the matching cost matrices before and after the training with the MC-CNN-WS on the KITTI 2012 dataset. The top figure shows left rectified stereo image with two highlighted epipolar lines. The pictures below show the matching cost matrices for these epipolar lines. The dark elements in the cost matrices correspond to the lower cost. WTA error before the training is 42.01%, and 14.61% after the training. Note that before the training we can not visually distinguish the best matches in the cost matrices, while after the training they are visible.

performance is the reason why we don't need to pre-train the network before applying the contrastive methods.

In Figure 3.5 we show failure cases of the deep matching cost after training with the proposed method. Most of the failures happen when the ground truth match is visually indistinguishable from the incorrect match picked by the network. This situation can arise when the reference patch is from a flat image area, an area with a repetitive texture, or an area with a horizontal edge. Notably, some failures are triggered by apparent errors in the ground truth. These errors worsen outcomes of the supervised learning as we show in § 3.3.7, but does not affect outcomes of our weakly supervised learning, since it does not use the ground-truth.

3.3.6 Weakly supervised domain adaptation

In this experiment, we study the ability of the proposed method to adapt a matching cost to a target domain without the ground truth. For that, we train the deep matching cost using the proposed method on one dataset and test on all datasets. The results are shown in Table 3.6. Notice that the accuracy is always better when the training and the validation examples come

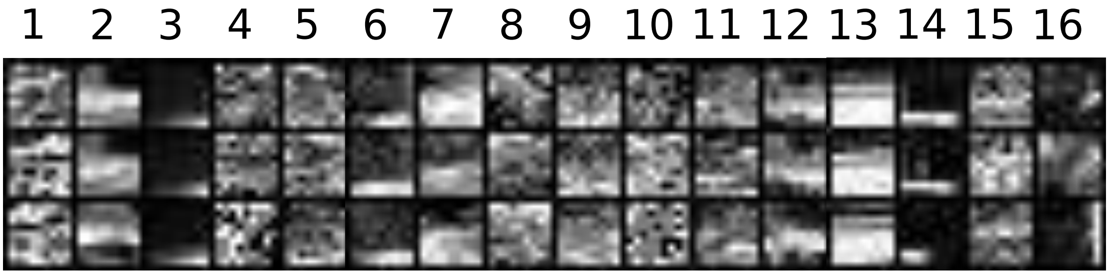


Figure 3.5 – Failure cases of the deep matching cost trained with the proposed method on the KITTI 2012 dataset. For each example, the three displayed patches from top to bottom correspond to: the reference patch, the predicted match, and the ground-truth match. Most of the incorrectly predicted matches are visually indistinguishable from the ground truth matches and correspond to the cases when the reference patch is from the area with the horizontal edge (3, 6, 13), flat image area (4, 5, 10), or area with a repetitive texture. Some failures are triggered by the apparent errors in the ground truth (2, 12, 14, 16).

Table 3.6 – Generalization error across datasets. The smallest WTA errors, shown in bold, correspond to the cases when the training and the validation sets are similar. This confirms that the proposed weakly supervised method can indeed tune the deep matching cost to a particular stereo system and dataset even without the ground truth disparity.

Training set	**Winner-take-all 3PE, [%]**		
	KITTI 2012	**KITTI 2015**	**Middlebury**
KITTI 2012	**13.90**	15.52	34.85
KITTI 2015	16.61	**14.08**	36.66
Middelbury	14.22	15.00	**29.60**

from the same population. This proves that the proposed weakly supervised method can to tune a deep matching cost to a particular stereo system without the ground truth.

3.3.7 Training with a noisy ground truth

In this experiment, we show that the proposed weakly supervised method has an advantage over the supervised baseline [19] in the presence of ground-truth noise. For that, we add noise to the sub-pixel accurate ground truth disparity of the Middlebury training set. We use two noise models: uniform, that simulates gross disparity errors, and Gaussian, that simulates sensor noise. To add uniform noise, we substitute the ground truth disparity with a sample from the uniform distribution $\mathscr{U}(0, d_{max})$ with some probability p_u. To add Gaussian noise, we simply add a sample from the Gaussian distribution $\mathscr{N}(0, \sigma_g^2)$ to the ground truth disparity.

From Table 3.7 it is clear that the supervised method underperforms in the presence of the noise. It is especially sensitive to the Gaussian noise, that is typical for depth sensors. This kind of noise is, for example, present in the KITTI datasets [119]. In contrast, the proposed method is unaffected by the noise since it does not use the ground truth.

Table 3.7 – Training in the presence of ground truth noise. We evaluate errors on the Middlebury validation dataset. In the presence of the ground truth noise, the performance of the supervised method plummets. In contrast, the proposed method is unaffected by the noise.

Method	**Ground truth noise**	**Winner-take-all 3PE, [%]**
MC-CNN fst [19]	No noise	29.94
	Uniform, $p_u = 5\%$	30.66
	Uniform, $p_u = 20\%$	31.35
	Gaussian $\mathcal{N}(0, 3^2)$	34.86
MC-CNN-WS (proposed)	–	**29.60**

3.4 Conclusion

We propose a novel weakly supervised method for training patch-matching cost for stereo reconstruction. This method allows training with a dataset without ground truth disparity, by relying on simple constraints coming from properties of a stereo system, and a rough knowledge about the scenes.

Benchmarking on standard datasets shows that the proposed method gives similar results as the baseline supervised method with the same network trained on the same but fully labeled datasets.

This good performance can be explained by the strong redundancy of fully labeled datasets, due to the continuity of surfaces, coupled with inevitable labeling errors. The latter can degrade the performance resulting from fully supervised training process, and could only be mitigated by using prior knowledge about the regularity of the labeling.

4 Applications-friendly deep stereo

Contents

4.1 Introduction **44**
4.2 Method **45**
    4.2.1 Matching module 46
    4.2.2 Sub-pixel maximum a posteriori estimator 48
    4.2.3 Sub-pixel cross-entropy loss 49
4.3 Experiments **50**
    4.3.1 Training on full-size images 50
    4.3.2 Effectiveness of the proposed estimator 51
    4.3.3 Effectiveness of the proposed loss 53
    4.3.4 Benchmarking 53
4.4 Conclusion **53**

This chapter is based on the following publication:

S. Tulyakov, A. Ivanov, and F. Fleuret. Practical Deep Stereo (PDS): Toward applications-friendly deep stereo matching. In *Proceedings of the international conference on Neural Information Processing Systems (NeurIPS)*, pages 5874–5884, 2018.

4.1 Introduction

End-to-end deep learning networks recently showed an extremely good performance for stereo matching. However, existing networks are difficult to use in practical applications since (1) they are memory-hungry and unable to process even modest-size images, (2) have disparity range fixed at training time. For some applications, such as satellite stereo imagery, these are serious problems since satellite images are very large, often reaching tens of megapixels, and have a variable baseline, depending on a time difference between stereo image acquisition. In this work, we propose Practical Deep Stereo (PDS) network that addresses these issues.

Main contributions of this work can be summarised as following:

1. We decrease the memory footprint of the network by introducing a novel bottleneck module. It compresses concatenated image descriptors into a compact matching signature before feeding them to the regularization module. Reduced memory footprint allows processing larger images and training on full-size images, that boosts network ability to utilize a large spatial context.

2. We propose to use for the inference a sub-pixel maximum a posterior (MAP) estimator approximation that computes an expectation around the disparity with minimum matching cost and thus robust to erroneous modes in the disparity distribution and allows to modify a disparity range without retraining. This is extremely practical, as it allows the community to develop reusable stereo networks, as they exist in image recognition (VGG, AlexNet, etc.). For training, we similarly introduce a sub-pixel criterion by combining the standard cross-entropy with a kernel interpolation, which provides faster convergence rates and higher accuracy.

3. The proposed network has better or comparable performance than the state-of-the-art methods on FlyingThings3D and KITTI datasets.

This chapter is organised as following:

First, in the “Method” section we describe the proposed deep learning-based method in details: in § 4.2.1 we introduce novel bottleneck matching module, drastically reducing memory footprint, in § 4.2.2 present sub-pixel MAP estimator approximation, insensitive to erroneous modes in a disparity distribution, and in § 4.2.3 propose a sub-pixel cross-entropy loss.

Next, in the “Experiments” section we validate our contributions: in § 4.3.1 we show that the reduced memory footprint allows training on full-size images and leveraging a large spatial context and improving the performance, in § 4.3.2 and § 4.3.3 demonstrate that the proposed estimator and the loss allow modifying the disparity range without re-training and improving the performance, in § 4.3.4 benchmark our method against the state-of-the-art baselines and show that it has the smallest 3-pixel-error (3PE) and the smallest or the second smallest mean absolute error (MAE) on the FlyingThings3D set, and ranked the third and the fourth on the KITTI 2015 and the KITTI 2012 sets respectively. Finally, in the “Conclusion” section we

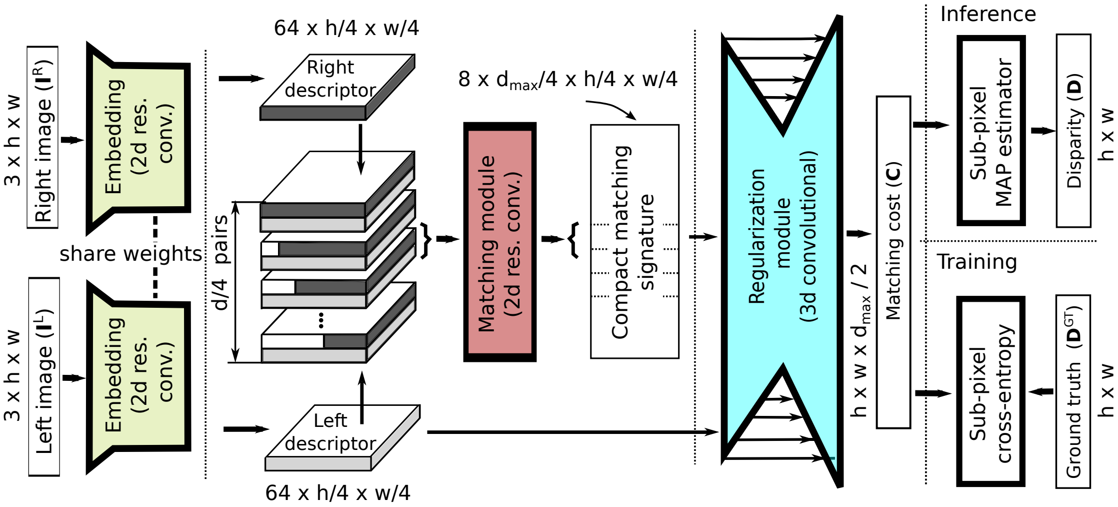

3 x h x w
Right image ($\mathbf{I}^R$)
Embedding (2d res. conv.)
share weights
3 x h x w
Left image ($\mathbf{I}^L$)
Embedding (2d res. conv.)
64 x h/4 x w/4
Right descriptor
d/4 pairs
Left descriptor
64 x h/4 x w/4
8 x d_{max}/4 x h/4 x w/4
Matching module (2d res. conv.)
Compact matching signature
Regularization module (3d convolutional)
h x w x d_{max} / 2
Matching cost (C)
Inference
Sub-pixel MAP estimator
Disparity (D)
h x w
Training
Sub-pixel cross-entropy
Ground truth ($\mathbf{D}^{GT}$)
h x w

Figure 4.1 – Functional structure of the proposed network and processing flow during training and inference. We outline the input and output quantities with thin lines and the processing modules with thick ones. The yellow shapes are *embedding* modules, the red rectangle is the *matching* module and the turquoise shape is the *regularization* module. The *matching* module is a contribution of this work. In the previous methods [75, 80], authors directly feed the concatenated left and shifted right descriptors to the regularization module (hourglass network). We represent 4d compact matching signature tensor as a 3d tensor by combining the feature indexes and disparities along the vertical axis.

summarize our results.

4.2 Method

The proposed network takes as input color stereo images $\{\mathbf{I}^L, \mathbf{I}^R\}$ each of size $3 \times h \times w$ and produces a *matching cost* tensor $\mathbf{C}$ as following

$$\mathbf{C} = \text{Net}(\mathbf{I}^L, \mathbf{I}^R \mid \mathbf{\Theta}, d_{max}), \tag{4.1}$$

where $\mathbf{\Theta}$ are the model's parameters, and d_{max} is a maximum disparity.

The computed matching cost tensor is such that $C_{d,y,x}$ is the cost of matching the pixel $I^L_{y,x}$ in the left image to the pixel $I^R_{y,x-d}$ in the right image, which is equivalent to assigning the disparity $D_{y,x} = d$ to the left image pixel. We can convert the matching cost tensor $\mathbf{C}$ into a posterior probability tensor as

$$P\left(\mathbf{D} \mid \mathbf{I}^L, \mathbf{I}^R\right) = \underset{d}{\text{softmin}}\left(C_{d,y,x}\right) \tag{4.2}$$

Note, that the size of the matching cost tensor is $\frac{d_{max}}{2} \times h \times w$, because we compute matching costs only for even disparities to reduce memory footprint. However, for clarity in all formulas we assume that the tensor has size $d_{max} \times h \times w$.

The modular architecture of the proposed network and processing flow during training and inference are shown in Figure 4.1, and we can summarize the input(s) and the output(s) of each module as follows

- The *embedding* module takes as an input a color image of size $3 \times h \times w$, and computes an *image descriptor* of size $c \times \frac{h}{4} \times \frac{w}{4}$.
- The *matching* module for each disparity d takes as an input the left and shifted right image descriptors, each of size $c \times \frac{h}{4} \times \frac{w}{4}$, and computes a *compact matching signature* $\frac{c}{8} \times \frac{h}{4} \times \frac{w}{4}$. This module is unique to our network and we describe it in details in § 4.2.1.
- The *regularization* module is an hourglass 3d convolution neural network with shortcut connections between the contracting and the expanding parts. It takes a tensor composed of concatenated compact matching signatures for all disparities of size $\frac{c}{8} \times \frac{d_{max}}{4} \times \frac{h}{4} \times \frac{w}{4}$, and computes a matching cost $\mathbf{C}$ of size $\frac{d_{max}}{2} \times h \times w$.

We provide additional information about every module such as convolution filter size and channels number in Table 4.1.

According to the taxonomy in [120], all traditional stereo matching methods consist of (1) *matching score computation*, (2) *score aggregation*, (3) *optimization*, and (4) *disparity refinement* steps. In the proposed network, the embedding and the matching modules are roughly responsible for the step (1) and the regularization module for the steps (2-4).

Besides the matching module, there are several other design choices that reduce the memory footprint of our network during training and inference. In contrast to [75], we use aggressive four-times sub-sampling in the embedding module, and our regularization module produces probabilities only for even disparities. Also, after each convolution and transposed convolution in the network, we place Instance Normalization (IN) [137] instead of Batch Normalization (BN) since during the training we use full-size images instead of mini-batches of image patches.

4.2.1 Matching module

The core of state-of-the-art methods [75, 76, 78, 80] is a 3d convolutional Hourglass network used as the regularization module, that takes as an input a tensor composed of concatenated left-right image descriptors for all possible disparity values. Because of the large size of this tensor, these methods have a huge memory footprint during the inference.

We decrease the memory usage by implementing a novel matching module with a “bottleneck” architecture, that we show in Figure 4.2. For each disparity, the matching module takes concatenated left and shifted right descriptor and produces a compact matching signature,

Table 4.1 – The architecture of the proposed network. Each Residual block consists of two 2d convolutions followed by a shortcut connection. Every convolution and transposed convolution, including these in the Residual blocks, is followed by LeakyReLU with a negative slope 0.2 and Instance Normalization [137]. We set $c = 64$.

Layer	**Layer Description**	**Output Dimension**
	color image	$3 \times h \times w$
	Embedding module	
E1	2d convolution $3 \times 5 \times 5 \times c$ stride 2	$c \times \frac{1}{2}h \times \frac{1}{2}w$
E2	2d convolution $c \times 5 \times 5 \times c$ stride 2	$c \times \frac{1}{4}h \times \frac{1}{4}w$
E3	2× Residual block with $c \times 3 \times 3 \times c$ 2d conv.	$c \times \frac{1}{4}h \times \frac{1}{4}w$
E4-redir.	2d convolution $c \times 3 \times 3 \times \frac{c}{8}$ no IN, LeakyReLU	$\frac{1}{8}c \times \frac{1}{4}h \times \frac{1}{4}h$
	Matching module	
M1	concatenate left-right embeddings E3	$2c \times \frac{1}{4}h \times \frac{1}{4}w$
M2	2d convolution $128 \times 3 \times 3 \times c$	$2c \times \frac{1}{4}h \times \frac{1}{4}w$
M3	2× Residual block with $c \times 3 \times 3 \times c$ 2d convolution	$c \times \frac{1}{4}h \times \frac{1}{4}w$
M4	2d conv. $c \times 3 \times 3 \times \frac{c}{8}$ no IN, LeakyReLU	$\frac{1}{8}c \times \frac{1}{4}h \times \frac{1}{4}w$
	Regularization module	
H1	concatenate joint embeddings M4	$\frac{1}{8}c \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
H2	3d convolution $\frac{c}{8} \times 3 \times 3 \times 3 \times \frac{c}{8}$	$\frac{1}{8}c \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
H3	3d convolution $\frac{c}{8} \times 3 \times 3 \times 3 \times \frac{c}{4}$, stride 2	$\frac{1}{4}c \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H4	H3 + E4-redir.	$\frac{1}{4}c \times \frac{1}{4}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H5	3d convolution $\frac{c}{4} \times 3 \times 3 \times 3 \times \frac{c}{4}$	$\frac{1}{4}c \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H6	H5 + H4	$\frac{1}{4}c \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H7	3d convolution $\frac{c}{4} \times 3 \times 3 \times 3 \times \frac{c}{2}$, stride 2	$\frac{1}{2}c \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
H8	3d convolution $\frac{c}{2} \times 3 \times 3 \times 3 \times \frac{c}{2}$	$\frac{1}{2}c \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
H9	H8 + H7	$\frac{1}{2}c \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
H10	3d convolution $\frac{c}{2} \times 3 \times 3 \times 3 \times c$, stride 2	$c \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
H11	3d convolution $c \times 3 \times 3 \times 3 \times c$	$c \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
H12	H11 + H10	$c \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
H13	3d convolution $c \times 3 \times 3 \times 3 \times 2c$, stride 2	$2c \times \frac{1}{64}d_{max} \times \frac{1}{64}h \times \frac{1}{64}w$
H14	3d trans. conv. $2c \times 4 \times 4 \times 4 \times c$, stride 2	$c \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
H15	H14+H11	$c \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
H16	3d convolution $c \times 3 \times 3 \times 3 \times c$	$c \times \frac{1}{32}d_{max} \times \frac{1}{32}h \times \frac{1}{32}w$
H17	3d trans. convolution $c \times 4 \times 4 \times 4 \times \frac{c}{2}$, stride 2	$\frac{1}{2}c \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
H18	H17+H8	$\frac{1}{2}c \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
H19	3d convolution $\frac{c}{2} \times 3 \times 3 \times 3 \times \frac{c}{2}$	$\frac{1}{2}c \times \frac{1}{16}d_{max} \times \frac{1}{16}h \times \frac{1}{16}w$
H20	3d trans. convolution $\frac{c}{2} \times 4 \times 4 \times 4 \times \frac{c}{4}$, stride 2	$\frac{1}{4}c \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H21	H20+H5	$\frac{1}{4}c \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H22	3d convolution $\frac{c}{4} \times 3 \times 3 \times 3 \times \frac{c}{4}$	$\frac{1}{4}c \times \frac{1}{8}d_{max} \times \frac{1}{8}h \times \frac{1}{8}w$
H23	3d trans. convolution $\frac{c}{4} \times 4 \times 4 \times 4 \times \frac{c}{8}$, stride 2	$\frac{1}{8}c \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
H24	H23+H3	$\frac{1}{8}c \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
H25	3d convolution $\frac{c}{8} \times 3 \times 3 \times 3 \times \frac{c}{8}$	$\frac{1}{8}c \times \frac{1}{4}d_{max} \times \frac{1}{4}h \times \frac{1}{4}w$
H26	3d trans. convolution $\frac{c}{8} \times 4 \times 4 \times 4 \times \frac{c}{16}$, stride 2	$\frac{1}{16}c \times \frac{1}{2}d_{max} \times \frac{1}{2}h \times \frac{1}{2}w$
H27	3d trans. convolution $\frac{c}{16} \times 3 \times 4 \times 4 \times 1$, stride (1,2,2), no IN, LeakyReLU	$\frac{1}{2}d_{max} \times h \times w$

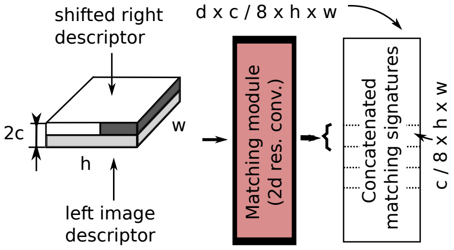
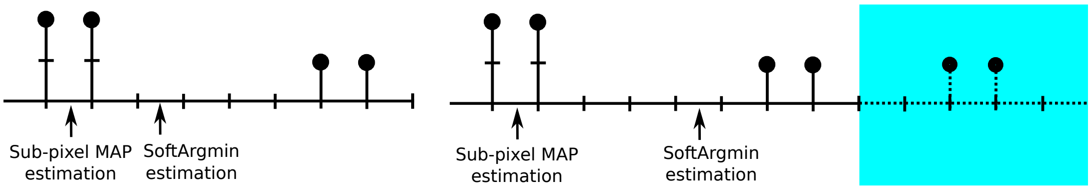


Figure 4.2 – For each disparity, the matching module takes concatenated left and shifted right descriptor and produces a compact matching signature, that is, similarly to the correlation matrix from [21] carry information about the goodness of match, but has multiple channels. Because the matching signature has fewer channels than the concatenated descriptors we reduce memory footprint during inference.



(a) Multi-modal distribution

(b) Distribution of extended disparity range

Figure 4.3 – Disadvantages of the SoftArgmin estimator: (a) in presence of the multi-modal distribution the SoftArgmin blends all the modes and produces the incorrect disparity estimate; (b) when the disparity range is extended (blue area), the SoftArgmin estimate degrades due to the additional modes.

that is, similarly to the correlation matrix from [21] carry information about the goodness of match, but has multiple channels. Then, we concatenate matching signatures for all disparities in 4d tensor and feed it to the regularization module. Because the matching signature has fewer channels than the concatenated descriptors we reduce memory footprint during inference. This contrasts with existing methods, which directly feed the concatenated descriptors [75, 76, 78, 80] to the regularization module. Reducing the memory footprint allows processing larger images during the inference, and consequently using a larger spatial context for solving matching ambiguities, which translates directly into better performance.

This module is inspired by CRL [77] and DispNetCorr1D [59, 74] which control the memory footprint (as shown in Table 4.6) by feeding correlation results instead of concatenated descriptors to the Hourglass network and by [63] which show the superior performance of joint left-right image embedding. We also borrow some ideas from the bottleneck module in ResNet [138], since it also encourages compressed intermediate representations.

4.2.2 Sub-pixel maximum a posteriori estimator

In state-of-the-art methods, a stereo network produces a matching cost and then use SoftArgmin module [75, 76, 78, 80], introduced in [75], to compute the predicted sub-pixel disparity. The name SoftArgmin comes from the fact that this function computes disparity of a match with the minimum matching cost in a "soft" way. In fact, the SoftArgmin is simply an expectation of a probability distribution over the disparity

$$\widehat{D}_{y,x} = \sum_{d} d \cdot P\left(D_{y,x} = d \mid \mathbf{I}^L, \mathbf{I}^R\right), \tag{4.3}$$

where $\widehat{D}_{y,x}$ is an estimated disparity for left image pixel with coordinates (y, x).

This SoftArgmin approximates a sub-pixel *maximum a posterior (MAP)* solution when the distribution over disparity is uni-modal and symmetric. However, as illustrated in Figure 4.3, this strategy suffers from two key weaknesses. First, when these assumptions are not fulfilled, for instance, if the distribution is multi-modal, the averaging blends the modes and produces the disparity estimate far from all of them. Second, if we extend the disparity range without retraining, the estimate degrades due to additional modes.

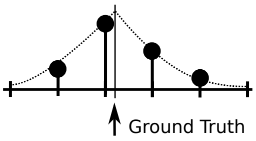


Figure 4.4 – Target distribution of sub-pixel cross-entropy is a discretized Laplace distribution centered at sub-pixel ground truth disparity.

The authors of [75] argue that a network adapts to the SoftArgmin during training by re-scaling its output values to make the distribution uni-modal. However, the network learns to perform the re-scaling only for the disparity range used during the training. If we decide to change the disparity range during the test, we will have to retrain the network.

To address these drawbacks, we propose to use a sub-pixel MAP approximation for the inference that computes using Equation 4.3 and a uni-modal posterior distribution calculated using only matching cost values around the disparity with the minimum matching cost $\tilde{D} = \arg\min_d C_{d,y,x}$ as follows

$$P(D_{y,x} \mid \mathbf{I}^L, \mathbf{I}^R) = \begin{cases} \operatorname{softmin}_{d:|d-\tilde{d}|\leq\Delta} C_{d,y,x} & \left|D_{y,x} - \tilde{D}\right| \leq \Delta \\ 0 & \text{otherwise} \end{cases}, \tag{4.4}$$

where Δ is an *estimator's support*, which we set equal to 2 based on a small scale grid search. The approximation works under the assumption that the disparity distribution is symmetric in the vicinity of a major mode.

Note, that in contrast to the SoftArgmin, the proposed sup-pixel MAP is used only for inference. During the training, we use the posterior disparity distribution computed using Equation 4.2 and the sub-pixel cross-entropy loss discussed in the next section.

4.2.3 Sub-pixel cross-entropy loss

Many deep stereo networks rely on L^1 loss [75, 76, 78, 80], even though the "natural" choice for classification by design networks, producing distribution over discrete disparity values is cross-entropy. The L^1 loss is often selected because it empirically [75] performs better than the cross-entropy and because when combined with SoftArgmin, it allows training a network with sub-pixel ground truth.

In this work, we propose a novel sub-pixel cross-entropy that provides faster convergence and better accuracy. The target distribution of the proposed loss is a discretized Laplace distribution centered at the ground truth disparity $D^{GT}_{y,x}$, shown in Figure 4.4 and computed as follows

$$\text{Laplace}(D_{y,x} = d \mid D^{GT}_{y,x}, b) = \frac{1}{N}\exp\left(-\frac{|d - D^{GT}_{y,x}|}{b}\right), \text{ where } N = \sum_{d}\exp\left(-\frac{|d - D^{GT}_{y,x}|}{b}\right), \tag{4.5}$$

where b is a *diversity* of the Laplace distribution, which we set equal to 4 pixels, reasoning that the distribution should cover at least several discrete disparities. With this target distribution, the cross-entropy is computed as usual:

$$\text{Loss}(\boldsymbol{\Theta}) = -\frac{1}{wh}\cdot\sum_{y,x}\sum_{d}\text{Laplace}(D_{y,x} = d \mid \mu = D^{GT}_{y,x}, b)\cdot\log(\text{softmin}(C_{d,y,x})) \tag{4.6}$$

The proposed sub-pixel cross-entropy is different from soft cross-entropy [136] since in the former case, in each discrete location, the target distribution is a smooth function of a distance to the sub-pixel ground truth. This allows training the network to produce a distribution from which we can compute sub-pixel disparities using the estimator, proposed in the previous section.

4.3 Experiments

We guarantee reproducibility of all experiments in this section by using only available datasets, and making our code available online[1].

We perform all experiments with the PyTorch framework [139]. The weighs and the biases of the network we initialize using default PyTorch initialization and train the network on each dataset as shown in Table 4.2. During the training, we normalize training images to zero mean and unit variance. We optimize the loss function with the RMSprop method with standard settings. In our experiments, we use the FlyingThings3D and the KITTI datasets and 3-pixels-error (3PE) and mean absolute error (MAE) accuracy measures described in Chapter 2. For validation sets, we withhold 500 images from the FlyingThings3D training set, and 58 from the KITTI training set, respectively.

4.3.1 Training on full-size images

In this section, we show the effectiveness of using full-size images during training and inference, which is possible due to a small memory footprint of the proposed network. For that we train our network till the convergence on the FlyingThings3D dataset with L^1 loss and

---

[1] https://github.com/tlkvstepan/PracticalDeepStereo_NIPS2018

Table 4.2 – Summary of training settings for the FlyingThings3D and the KITTI datasets.

	FlyingThings3D	**KITTI**
Mode	from scratch	fine-tune
Learning schedule	0.01 for 120k, ×0.5 every 20k	0.005 for 50k, ×0.5 every 20k
Number of iterations	160k	100k
Training image size	960 × 540 (full-size)	1164 × 330
Maximum disparity	255	255
Augmentation	not used	mixUp [140], zoom and crop

Table 4.3 – Disparity error of the proposed network on the FlyingThings3D test set as a function of training patch and test patch sizes. Interestingly, even the network trained on small image patches performs better when provided with access to a larger spatial context during inference. As expected, the network trained on full-size images, highlighted in the table, performs the best. Note, that during the training we use SoftArgmin with L^1 loss.

Training patch size	**Test patch size**	**3-pixel-error, [%]**	**Mean absolute error, [px]**
512 × 256	512 × 256	8.63	4.18
512 × 256	960 × 540	5.28	3.55
960 × 540	960 × 540	4.50	3.40

SoftArgmin twice, the first time we use 512 × 256 training patches randomly cropped from training images as in [75, 80], and the second time, we use full-size 960 × 540 training images.

As shown in Table 4.3, even the network trained on small image patches performs better when provided with access to a larger spatial context during inference. As expected, the network trained on full-size images makes better use of the said context and performs significantly better.

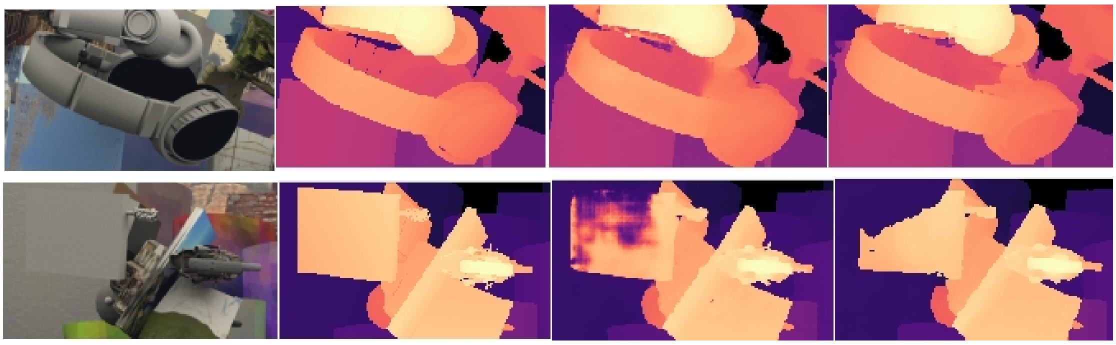
4.3.2 Effectiveness of the proposed estimator

In this section, we show the advantages of the proposed estimator over the SoftArgmin. For that, we train the proposed network till convergence on the FlyingThings3D with the SoftArgmin, L^1 loss and full-size training images and test it twice: the first time with the SoftArgmin for inference, and the second time with the proposed sub-pixel MAP. As shown in Table 4.4, the substitution leads to the reduction of the 3-pixels-error (3PE) and the slight increase of the mean absolute error (MAE). The latter probably happens because in an erroneous area SoftArgmin's estimates are wrong, but closer to ground truth since it blends all distribution modes, as shown in Figure 4.5.

When, we test the same network with the disparity range increased from 255 to 511 pixels, the performance of the network with the SoftArgmin plummets, while the performance of the network with the sub-pixel MAP remains almost the same as shown in Table 4.4. This proves that the sub-pixel MAP allows modifying the disparity range of a network on-the-fly.

Table 4.4 – Effectiveness of the proposed sub-pixel MAP estimator on the FlyingThings3D set. Note, that when during the test we substitute the SoftArgmin with the proposed Sub-pixel MAP, highlighted in the table, we get lower 3PE and similar MAE. Moreover, when we increase disparity range twice, the MAE and the 3PE of the network with the Sub-pixel MAP, highlighted in the table, almost do not change, while the errors of the network with the SoftArgmin increase.

Estimator	**3-pixels-error, [%]**	**MAE, [px]**
Standard disparity range $\in [0,255]$		
SoftArgmin	4.50	3.40
Sub-pixel MAP	4.22	3.42
Extended disparity range $\in [0,512]$		
SoftArgmin	5.20	3.81
Sub-pixel MAP	4.27	3.53



(a) Left image (b) Ground truth (c) SoftArgmin (d) Sub-pixel MAP

Figure 4.5 – Examples of disparity estimation error with the SoftArgmin and the sub-pixel MAP on the FlyingThings3D set. Note that the SoftArgmin estimate (c), though wrong, is closer to the ground truth (b) than the sub-pixel MAP estimate (d). This explains the larger MAE of the sub-pixel MAP estimate.

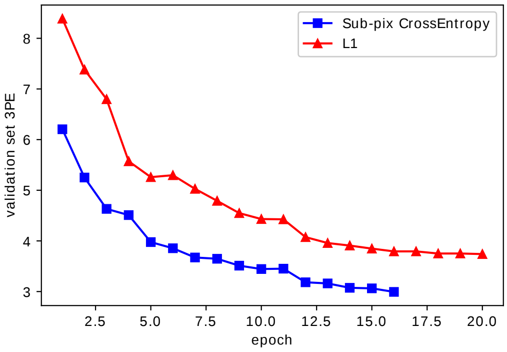


Figure 4.6 – Dynamics of 3-pixel-error on the validation set during training on the FlyingThings3D set with the proposed sub-pixel cross entropy (blue) and L^1 (red) criterion. Note, that with the proposed criterion the validation error converges faster and reaches the lower error than with L^1. Interestingly, [75] also reports faster convergence with one-hot cross-entropy than with L^1 loss, but contrary to our results, concludes that it results in a larger 3PE.

Table 4.5 – Effectiveness of the proposed sub-pixel cross-entropy loss on the FlyingThings3D set. As highlighted in the table, when we train the network with the sub-pixel cross-entropy loss, it has much lower 3PE and only slightly worse MAE than the network trained with the L^1 loss.

Loss	**3-pixels-error, [%]**	**MAE, [px]**
L^1 + SoftArgmin	4.22	3.42
Sub-pixel cross-entropy	3.80	3.63

4.3.3 Effectiveness of the proposed loss

In this section, we show the advantages of the proposed loss over the L^1. For that, we train the network with the proposed sub-pixel cross-entropy loss and compare it to the network trained with the SoftArgmin and L^1 loss. As shown in Table 4.5, the former network has much smaller 3PE and only slightly larger MAE. The convergence speed with the sub-pixel cross-entropy is also much faster than with L^1 loss as shown in Figure 4.6. Interestingly, [75] also reports faster convergence with one-hot cross-entropy than with L^1 loss, but contrary to our results, concludes that training with L^1 results in a larger 3PE.

4.3.4 Benchmarking

In this section, we show the effectiveness of the proposed method, compared to the state-of-the-art methods on the KITTI and the FlyingThings3D datasets. For the KITTI datasets, we compute disparity maps for the test sets with withheld ground truth and upload the results to the evaluation web site. For the FlyingThings3D set, we evaluate the performance on the test set ourselves as explained in Chapter 2.

FlyingThings3D benchmarking results are shown in Table 4.6. Notably, the proposed method has the lowest 3PE error according to both evaluation protocols and the lowest or the second lowest MAE, depending on the protocol. Moreover, in contrast to other methods, our method has a small memory footprint, number of parameters, and allows changing the disparity range without retraining.

KITTI 2012 and KITTI 2015 benchmarking results are shown in Table 4.7. The proposed method ranks the third on the KITTI 2015 set and the fourth on the KITTI 2012 set, taking into account state-of-the-art results not yet officially published at the time of the writing, such as iResNet-i2 [79], PSMNet [80] and LRCR [78] methods.

4.4 Conclusion

In this work, we address two issues precluding the use of deep networks for stereo reconstruction in many practical situations in spite of their excellent accuracy: their large memory footprint, and the inability to adjust to a different disparity range without retraining. We show

Table 4.6 – FlyingThings3d benchmark snapshot from 15/05/2018 with top methods, including not yet officially published at the time of the writing iResNet-i2 [79], PSMNet [80], and LRCR [78]. Besides the errors, we also show the number of parameters and inference memory footprint for each method. DispNetCorr1D [74], CRL [77], iResNet-i2 [79] and LRCR [78] predict disparities as classes and are consequently over-parameterized. GC [75] omits an explicit correlation step, which results in the large memory usage during inference. The proposed PDS network, highlighted in the table, has a small number of parameters and memory footprint, the smallest 3PE and the smallest or the second smallest MAE, depending on the evaluation protocol, and it is the only method able to handle different disparity ranges without retraining. Note, that for our method we report two results. The result outside of brackets is obtained using the protocol of the PSM [80] method, according to which the errors are calculated only for ground truth pixel with disparity < 192. The result in the brackets is calculated according to the protocol of the CRL [77], DispNetCorr1D [74] and iResNet-i2 [79] methods, according to which the error is calculated only for images where less than 25% of pixels have disparity > 300, as explained in [77]. Inference memory footprints are our theoretical estimates based on network structures and do not include memory required for storing networks' parameters (real memory footprint will depend on implementation). Error rates and numbers of parameters are taken from the respective publications.

Method	**Parameters, [M]**	**Memory, [GB]**	**3PE, [%]**	**MAE, [px]**	**Modifiable Disparity**
PDS (proposed)	2.2	0.4	3.38 (2.89)	1.12 (0.87)	✓
PSM [80]	5.2	0.6	n/a	1.09	✗
CRL [77]	78	0.2	6.20	1.32	✗
iResNet-i2 [79]	43	0.2	4.57	1.40	✗
DispNetCorr1D [74]	42	0.1	n/a	1.68	✗
LRCR [78]	30	9.0	8.67	2.02	✗
GC [75]	3.5	4.5	9.34	2.02	✗

that by carefully revising conventionally used networks architecture to control the memory footprint and adapt analytically the network to the disparity range, and by using the new loss and the estimator to cope with a multi-modal posterior distribution and sub-pixel accuracy, it is possible to resolve these practical issues and reach state-of-the-art performance.

Table 4.7 – KITTI 2015 (top) and KITTI 2012 (bottom) benchmarks snapshot from 15/05/2018 with top-10 methods, including not yet officially published at the time of the writing iResNet-i2 [79], PSMNet [80], and LRCR [78]. Our method, highlighted in the table, is the third in the KITTI 2015 and the fourth in the KITTI 2012 leader boards. The tables are shown in default mode.

Rank	Submission date	Method	3PE (all pixels), [%]	Time, [s]
1	30/12/2017	PSMNet [80]	2.16	0.4
2	18/03/2018	iResNet-i2 [79]	2.44	0.12
3	15/05/2018	PDS (proposed)	2.58	0.5
4	24/03/2017	CRL [77]	2.67	0.47
5	27/01/2017	GC-NET [75]	2.87	0.9
6	15/11/2017	LRCR [78]	3.03	49
7	15/11/2016	DRR [73]	3.16	0.4
8	08/11/2017	SsSMnet [76]	3.40	0.8
9	15/12/2016	L-ResMatch [72]	3.42	48
10	26/10/2015	Displets v2 [40]	3.43	265

Rank	Submission date	Method	3PE (non-occluded), [%]	Time, [s]
1	31/12/17	PSMNet [80]	1.49	0.4
2	23/11/17	iResNet-i2 [79]	1.71	0.12
3	27/01/17	GC-NET [75]	1.77	0.9
4	15/05/18	PDS (proposed)	1.92	0.5
5	15/12/16	L-ResMatch [72]	2.27	48
6	11/09/16	CNNF+SGM [140]	2.28	71
7	15/12/16	SGM-NET [71]	2.29	67
8	08/11/17	SsSMnet [76]	2.30	0.8
9	27/04/16	PBCP [132]	2.36	68
10	26/10/15	Displets v2 [40]	2.37	265

5 Dense deep event-based stereo

Contents

5.1 Introduction **58**
5.2 Background **59**
  5.2.1 Event-based stereo 60
  5.2.2 Deep learning with event sequences 61
5.3 Method **62**
  5.3.1 Events sequence embedding 63
5.4 Experiments **64**
  5.4.1 Dataset 67
  5.4.2 Importance of spatial and temporal context 67
  5.4.3 Effectiveness of the temporal aggregation 68
  5.4.4 Benchmarking 69
  5.4.5 What does continuous fully-connected layer learn? 70
5.5 Conclusion **73**

This chapter is based on the following publication:

S. Tulyakov, F. Fleuret, M. Kiefel P. Gehler and M. Hirsch. Learning an event sequence embedding for dense event-based deep stereo. *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019 (To appear).

5.1 Introduction

Today, a frame-based camera is the sensor of choice for stereo vision. However, these cameras, originally developed for acquisition of static images rather than for sensing of dynamic uncontrolled visual environments, suffer from high power consumption, data rate, latency, and low dynamic range. An *event-based image sensor* addresses these drawbacks by mimicking a biological retina. Instead of measuring the intensity of every pixel in a fixed time-interval, it reports events of significant pixel intensity changes. However, asynchronous event sequences require special handling, since the best performing deep methods for stereo reconstruction work only with synchronous, spatially gridded data. We address this problem in this work.

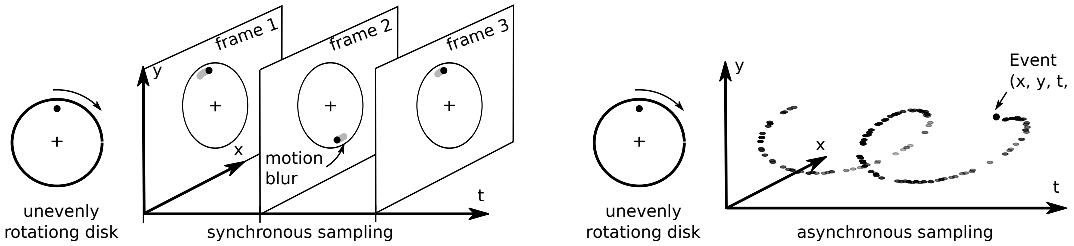
Main contributions of this work can be summarized as following

1. We propose a new learning-based module for an event sequence embedding, for use in different applications. The module builds a representation of an event sequence by firstly aggregating information locally across time, using a novel fully-connected layer for an irregularly sampled continuous domain, and then across discrete spatial domain.
2. We use this embedding to design the first deep network-based method for event-based stereo reconstruction. The method is based on an architecture with a large receptive field that uses large context and allows stereo reconstruction for locations without events. We demonstrate that this method significantly outperforms other competing approaches on the Multi Vehicle Stereo Event Camera Dataset (MVSEC).

This chapter is organised as follows

First, in the “Background” section we provide information about event-based camera and methods required for the understanding of this work: in § 5.2.1 we review stereo matching methods for an event-based camera and highlight major research directions, and in § 5.2.2 overview hand-crafted embedding methods for event sequences used in deep learning methods.

Next, in the “Method” section we propose a deep learning-based stereo matching method for an event-based camera based on novel event sequence embedding, which we introduce in § 5.3.1. Then, in the “Experiments” section we validate contributions of this work: in § 5.4.1 we introduce the dataset and the evaluation protocol, in § 5.4.2 discuss importance of temporal and spatial context, in § 5.4.3 compare the proposed temporal aggregation method to the hand-crafted baseline, in § 5.4.4 show that the stereo matching method, based on the proposed temporal aggregation outperforms the state-of-the-art methods by a large margin, in § 5.4.5 visualize filters of the proposed continuous fully-connected layer. Finally, in the “Conclusion” section we summarize this work.



(a) Conventional sensor (a) Event-based sensor

Figure 5.1 – Working principles of a conventional and an event-based sensor. The conventional frame-based sensor (a) acquire still pictures synchronously, at a fixed frame rate. In contrast, the event-based sensor (b) records asynchronous events of significant pixel intensity changes. Every event is characterized by a discrete position (x, y), timestamp t, which is so precise that it can be considered continuous, and binary polarity δ, which depends on the sign of the intensity change (*i.e.* “dark to bright” or “bright to dark”).

5.2 Background

Conventional frame-based sensor acquires still pictures at a fixed time interval or frame rate. In contrast, the retina in a human eye operates on completely different principles. Nobel prize winning experiments [141] showed that the retina is most sensitive to temporal brightness gradients, and is blind to static scenes in the absence of eye movements [142]. These principles inspired the development of event-based image sensors [143, 144].

In an event-based image sensor, pixels are sensitive to temporal brightness contrast and trigger binary events with a rate proportional to the temporal gradient of photo-current. Events can have positive or negative polarity depending on the sign of the gradient (*i.e.* “dark to bright” or “bright to dark”). Triggered events are recorded by the sensor asynchronously with information about their spatial positions, polarities, and timestamps accurate to microseconds. Working principles of the conventional frame-based and an event-based sensor are illustrated in Figure 5.1.

Event-based image sensors have several advantages over the conventional frame-based sensors. First, they have higher dynamic range and thus do not saturate in extreme lighting conditions, such as bright daylight and night with minimum illumination, thanks to pixel-wise gain and integration time control. Secondly, they have lower power consumption and data rate, since they only transmit information about significant brightness changes, and thus can be used in power-constrained systems. Finally, due to immediate transmission of every triggered event with a microsecond-accurate timestamp, such sensors have lower latency and can be used in time-critical applications. However, these advantages come at a cost. Event-based sensors have lower resolution, because their pixels are more complex, and do not provide rich intensity information. We summarize advantages and drawbacks of event-based image sensors in Table 5.1.

Unique properties of event-based image sensors make them attractive for low-latency dynamic

Table 5.1 – Advantages and drawbacks of an event-based image sensor, such as [144], compared to a conventional frame-based sensor. In the table, we show orders of magnitude for every characteristic, rather than precise values and highlight advantages of event-based sensors.

Characteristic	**Conventional sensor**	**Event-based sensor**
Dynamic range, [dB]	50	130
Power consumption, [W]	1	0.01
Data rate, [Mb/s]	100	0.1
Latency, [ms]	10	0.001
Resolution, [MP]	1	0.01
Intensity information	✓	✗

vision applications in environments with uncontrolled illumination, such as tracking [145], robot control [146], or object recognition [147]. Depth estimation, that we investigate in this work, could drastically improve performance in these applications, and open the way to novel use cases, such as augmented reality.

5.2.1 Event-based stereo

Due to the novelty of event-based image sensors, only a few event-based stereo methods have been proposed, none of which is learning-based.

One line of research investigates how to represent and compare events. This is a hard problem because events have a few spatio-temporal neighbors and only one binary feature. Early methods [148, 149] compared events using only their timestamps which led to matching ambiguities, due to the noise, variable cameras sensitivity, and imperfect camera synchronization. Therefore, later methods relied on hand-crafted descriptors [150–152] and similarity measures [153–155]. In [150] descriptors are computed as a bank of orientation-sensitive spatial filter responses, in [151] as a vector of distances to closest events in several spatial directions, in [152] as a histogram of orientations of vectors pointing to the closest events in a spatial window. As for similarity measures, spatial windows with events are compared in [153] using average distances to the closest events, in [155] using average inverse timestamp difference between corresponding events, and in [154] using intersection-over-union of events.

The second research direction explores how to apply regularization in stereo matching. This is a challenging problem due to the sparsity of data in both time and space, which consequently cannot be represented with conventional Markov Random Field (MRF) models. Some works [155–157] adopt heuristic cooperative regularization from [33] by defining a spatio-temporal inhibitory and excitatory neighborhood for each event, while others [158, 159] use belief propagation and semi-global matching on sparse MRF models, were nodes are active only during a fixed interval after receiving an event.

The third line of research explores how to accumulate events over the time to cope with the fact of individual events being noisy and not very informative, though leading to the undesirable effect of blurring object boundaries when using long accumulation times. Most methods use fixed accumulation intervals [148–150, 153], while [152] sets accumulation time equal to the average of the inverse of the event rate, and [154] warps event positions as if they were all triggered at the same time using depth hypothesis and known camera motion.

Another challenging direction is dense stereo matching using sparse event data. While most of the methods produce sparse disparity, authors in [160] reconstruct semi-dense disparity by fusing information from several view-points using known camera motion, [152] computes disparity at every location without an event by fitting a plane to its neighbor disparities.

Finally, a lot of works is dedicated to implementing stereo matching on neuromorphic chips and field-programmable gate arrays (FPGA) [161, 162].

To sum-up, all existing methods use hand-crafted event representations and grid-based image models, that support only simplistic priors, such as smoothness. Meanwhile, most successful frame-based stereo methods use learning-based representations [66], an area-based regularization [75] using deep networks, and even use monocular depth cues [104, 106]. This motivates our work, that focuses on developing end-to-end deep learning for event-based stereo matching.

5.2.2 Deep learning with event sequences

Event sequences can be processed using convolutional neural networks (CNNs), recurrent neural networks (RNNs) and specialized networks for event data.

One option is to use off-the-shelf CNNs, which are extremely successful in frame-based image processing. However, one problem is that convolutional modules work with dense images, where each pixel lies in a 2d or 3d (in case of video) discrete space, with an intensity value assigned to it. In contrast, an event sequence consists of sparse events, each lying in 3d space, with two discrete spatial dimensions, and one continuous temporal dimension, featuring one binary variable (polarity). Therefore, such a sequence needs to be transformed to a frame-based representation before it can be input to a standard CNN. Note, however, that naive binning of the time dimension is problematic since it would lose temporal information or produce tensors with a prohibitively large temporal dimension.

Existing methods [146, 163–167] use hand-crafted transformations to convert event sequences to frame-based representations, that we call *event images*. For example, [163] saves the polarity of the latest event, [164] sums event polarities in every location during a predetermined time interval, [146] counts the number of positive and negative events to avoid information loss due to polarity cancellation. To preserve time information, in addition to positive and negative event counts [166] saves the timestamp of the last positive and negative events at every location, while [167] saves the average timestamps of the updates. To capture time dynamic, [165, 167] stack several event images described earlier, for consecutive time intervals. The main drawback of all these representations is that they lose precise timing information about

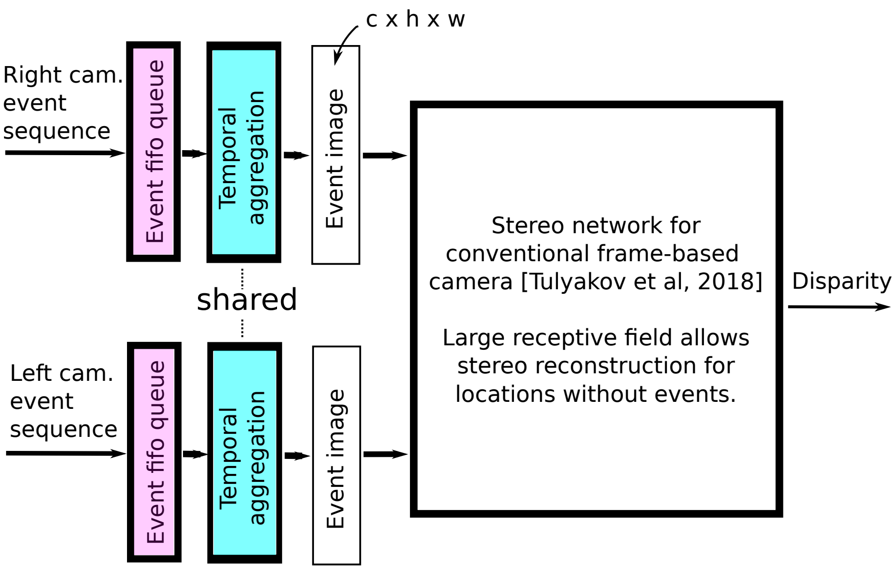

c x h x w
Right cam. event sequence
Left cam. event sequence
Event fifo queue
Temporal aggregation
Event image
shared
Stereo network for conventional frame-based camera [Tulyakov et al, 2018]
Large receptive field allows stereo reconstruction for locations without events.
Disparity

Figure 5.2 – Functional structure of the event-based stereo network. It is based on our network for frame-based stereo [173]. Its large receptive field allows performing stereo matching even for locations without events. To accommodate event sequences, we add *event queue* module which stores several most recent events in every location and learning-based *temporal aggregation* module which transforms an event sequence into the *event image*.

events.

Another seemingly natural choice, given the sequential nature of the data, is to use RNNs. For example, [168] uses RNNs with long-term memory for event-based recognition. Their main drawback is that they do not preserve the spatial information, that is crucial for some applications, such as stereo matching. Note, that convolutional RNNs, such as [169], preserve spatial information, but cannot be used for the same reason as CNNs.

Finally, one can use asynchronous networks, where every neuron has an internal state that is updated by events, such as Spiked Neural Networks (SNN) [170] or specially designed convolutional neural networks [171, 172]. Unfortunately, it is hard to build and train such networks, because they are not easily differentiable, and additionally difficult to implement in a standard framework that enables the use of available computational back-ends such as GPUs.

5.3 Method

The proposed network takes as an input left and right event sequences E^l, E^r, each consisting of n events sorted by the time of arrival $E = ((x_i, y_i, t_i, \delta_i) \mid t_{i+1} > t_i)_{i=1\dots n}$. Each event is a point in a three dimensional space with two discrete spatial coordinates and one continuous temporal coordinate $(x, y, t) \in [0\dots w) \times [0\dots h) \times \mathbb{R}$ and one polarity feature $\delta \in \{-1, 1\}$, where w and h correspond to the width and height of an image sensor, measured in pixels. Given

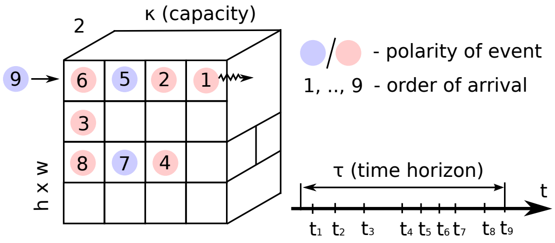


Figure 5.3 – The *event queue* is a 4d tensor of size $2 \times \kappa \times h \times w$. It stores $\leq \kappa$ most recent events at each location and keeps each event for at most τ seconds. The figure depicts the queue with width and height fused to a single dimension. When a new event (9) arrives, it pushes older events (6, 5, 2) to the end of the queue and the oldest event (1) out of the queue and occupies the first position.

such event sequences, the network computes an *estimated disparity* tensor $\widehat{\mathbf{D}}$ as

$$\widehat{\mathbf{D}} = \mathrm{Net}(E^l, E^r \mid \mathbf{\Theta}, d_{max}) \in [0, d_{max}]^{h \times w}, \tag{5.1}$$

where $\mathbf{\Theta}$ are network parameters and d_{max} is a maximum disparity.

The functional structure of the network is shown in Figure 5.2. It is based on our network for frame-based stereo [173] described in details in Chapter 4. Its large receptive field allows performing stereo matching even for locations without events. To accommodate event sequences, we add *event queue* module which stores several most recent events in every location and learning-based *temporal aggregation* module which transforms an event sequence to an event image. These modules are the main novelties of this work and they will be described in details in the next section.

5.3.1 Events sequence embedding

In this work, we focus on a special family of embedding functions that can be represented as a composition $G_S(G_\tau(\cdot))$ of two functions: *temporal aggregation* $G_\tau(\cdot)$ and *spatial aggregation* $G_S(\cdot)$. The *temporal aggregation* function is defined per-location and it takes a *local event sequence* $E(x,y) = ((x_i, y_i, t_i, \delta_i) \in E \mid (x_i, y_i) = (x, y))$ as input, and produces an *event image* $\mathbf{I}$ of size $c \times h \times w$, as $I_{y,x} = G_\tau(E(x,y))$. The *spatial aggregation* is a translation-invariant function that is applied to sub-windows of the event image and produces an event descriptor $\mathbf{F}$ such that $\forall y, x,\ F_{y,x} = G_S\left(\mathbf{I}_{y-\Delta:y+\Delta, x-\Delta:x+\Delta}\right)$.

The spatially gridded nature of the event image allows using standard 2d convolutions. Therefore, throughout our experiments, we use embedding module from [173] for spatial aggregation and focus on developing a temporal aggregation method.

To implement various temporal aggregation methods, we need a way to efficiently accumulate events in each location. For that, we propose to use a First-In First-Out (FIFO) queue shown in Figure 5.3. It stores $\leq \kappa$ most recent events at each location and keeps each event for at most τ seconds. We call κ the *capacity* and τ the *time horizon* of the queue. In real applications, this queue can be efficiently implemented using a linked list or a simpler circular buffer. Note, that this queue works well regardless of amount of motion: in presence of fast motion, when events are frequent, it stores only the recent ones, while in presence of slow motion, when events are rare, it preserves old ones. We prune events that arrived more than τ seconds ago from the queue and replace them with zeros before applying the temporal aggregation.

Hand-crafted. In §5.2.2, we reviewed existing methods for converting event sequences to event images. All of them are hand-crafted temporal aggregation functions. One of these methods produces an event image by counting the number of positive and negative events and recording timestamps of the most recent positive and negative events at every location. Because similar methods [146, 163–167] worked well in many applications, we use this method as a baseline.

Continuous fully-connected layer (CFC). Ideally, however, we would like to take into account the fact that the events are sampled irregularly and have continuous timestamps. To do so, we use a *continuous fully-connected layer (CFC)*, where continuous kernels are themselves approximated by a multi-layer perceptron (MLP), that we call a *kernel network*. This network allows modeling arbitrary complex, continuous kernels and can be trained end-to-end along with the rest of the architecture. The overall idea is illustrated in Figure 5.5.

Let us compare the proposed layer to a standard fully-connected (FC) layer, to appreciate the differences. Given event polarities $\boldsymbol{\delta} = [\delta_1, \delta_2, \delta_3, \delta_4, \delta_5, 0, 0]$ for some location stored in the event queue, a single output of the conventional FC layer is computed as $I = \sigma\left(\sum_i^7 w_i \delta_i + b\right)$, where $\mathbf{w}$ is a weights vector and b is a bias and $\sigma(\cdot)$ is a non-linearity. In contrast, a single output of the proposed CFC layer is computed as $I = \sigma\left(\frac{1}{5}\sum_i^5 w(t_i) \cdot \delta_i + b\right)$. Note that as shown in Figure 5.4, for a standard FC layer, the weight of each polarity simply depends on the events order i, while for the proposed CFC, the weight is a continuous parametric function $w(t_i) = \text{KernelNet}(t_i)$, of real-valued event timestamp t_i. This allows to embed event sequences with irregularly spaced time intervals between events.

Note that a similar idea is used in [174]. However, in [174] authors use continuous convolutional layers, while we propose continuous fully-connected layer, moreover, they work with 3d points in a Euclidean space acquired by a LIDAR.

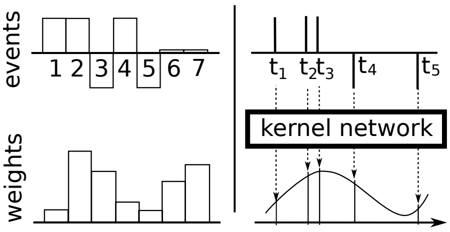
5.4 Experiments

All experiments in this section are done using the PyTorch framework [139] on publicly available datasets, and our code is available online [1].

In our experiments, besides the PDS [173] stereo network with a large receptive field, we use the network with a small 9 × 9 receptive field. This network is similar to "accurate network" from [19] with four 3 × 3 convolutional layers with 64 features for embedding and two 1 × 1 convolutional layers with 128 features for computing matching cost and we refer to it as MC-

---

[1] https://github.com/tlkvstepan/event_stereo_ICCV2019



(a) FC (b) CFC

Figure 5.4 – Comparison of the conventional fully-connected (FC) layer (a) to proposed continuous fully-connected (CFC) layer (b). In contrast to the FC, the CFC allows embedding event sequences with irregularly spaced time intervals between events.

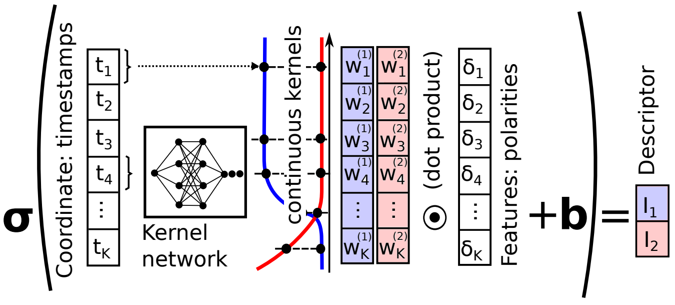


Figure 5.5 – In the *continuous fully-connected layer* depicted here, for every timestamp the kernel network computes two weights, and for all timestamps two weight vectors, each corresponding to a continuous kernel. To get an event sequence descriptor we multiply each of these vectors by the polarity vector using dot product, add bias and apply non-linearity. The corresponding weight vector, descriptor element and continuous kernel are shown in same color.

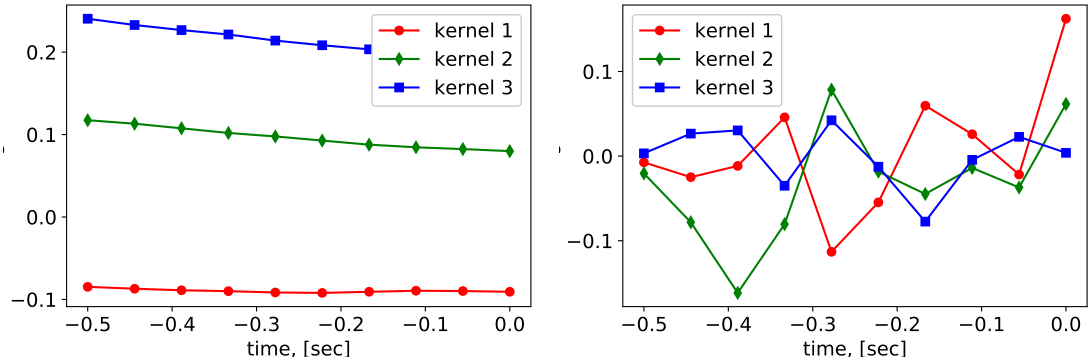
Table 5.2 – The architecture of the kernel network in the continuous fully-connected layer. The network takes as an input event timestamps of size $1 \times \kappa \times h \times w$ from the event queue and produces a weights tensor of size $64 \times \kappa \times h \times w$. By using 3d convolutions with kernel $1 \times 1 \times 1$ instead of a fully-connected layer we can process the entire event queue.

Layer Description	**Output Size**
3d convolution $1 \times 1 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times \kappa \times h \times w$
$2 \times$ 3d convolution $64 \times 1 \times 1 \times 1 \times 64 \circ$ ReLU	$64 \times \kappa \times h \times w$
3d convolution $64 \times 1 \times 1 \times 1 \times 64$	$64 \times \kappa \times h \times w$

CNN. We use this network to study the effectiveness of the proposed temporal aggregation in the absence of a large spatial context.

In our experiments, in the continuous fully-connected layer we use the kernel network with the architecture shown in Table 5.2, that was discovered using coarse grid search.

All networks are initialized using a default PyTorch initialization, except the kernel network, for which we develop the custom initialization described below. Usually network weights are initialized using a normal distribution $\mathbb{N}\left(0, \frac{2}{N_l + N_{l-1}}\right)$, where N_{l-1} and N_l are the numbers of inputs and outputs respectively. This initialization is called Xavier initialization [175] and it ensures that the variances of network activations and parameter gradients are kept constant across all layers. Since the kernel network essentially produces weights of the continuous fully-connected layer, we initialize its parameters such that its output is normally distributed. This is done computationally by sampling weights of the continuous fully-connected layer for timestamps $t_1, t_2, \dots t_M$ from a normal distribution $\mathbf{W} = [\mathbf{w}(t_1), \dots, \mathbf{w}(t_M)] \sim \mathbb{N}\left(0, \frac{2}{N_l + N_{l-1}}\right)$ and fitting the kernel network to these weights. Besides keeping the variances in check, this



(a) Before initialization (b) After initialization

Figure 5.6 – The continuous kernels of the kernel network before (a) and after (b) the custom initialization. Note, that before the initialization (a) the continuous kernels are mostly linear and very similar to each other, whereas after the initialization (b) they have complex and diverse shapes. For clarity, we show 3 kernels out of 64.

Table 5.3 – Optimal learning rate and event queue parameters for each combination of a temporal aggregation and stereo network selected using a "coarse" grid search on the validation set of the first split of the Indoor Flying set.

	Learning rate	**Capacity, [#]**	**Time horizon, [seconds]**
Small spatial context (MC-CNN [19])			
Hand-crafted	10^{-3}	all events	0.1
Continuous fully-connected	10^{-3}	15	0.5
Large spatial context (PDS [173])			
Hand-crafted	0.1	all events	0.1
Continuous fully-connected	10^{-4}	1	0.5

initialization ensures diversity of the resulting continuous kernels as shown in Figure 5.6. The biases of the fully-connected continuous layer are initialized with zeros.

The optimization is performed using RMSprop method with standard settings. For the training, we use full-sensor event sequences without augmentation. We consider only ground truth at locations corresponding to the most recent 15'000 events, if not explicitly stated otherwise. In all experiments, we normalize event polarities to $\mathcal{N}(0,1)$ and subtract the timestamp of the most recent event in the sensor from all timestamps.

For each combination of the temporal aggregation and stereo network, we chose the best learning rate, event queue capacity and time horizon shown in Table 5.3 using a "coarse" grid search on the validation set of the first split of the Indoor Flying set. In the case of the continuous fully-connected layer combined with MC-CNN, the larger event queue capacity might lead to even better result but we do not explore the parameter space further due to the time limitation.

5.4.1 Dataset

In this work, we use the Multi-Vehicle Stereo Event Camera Dataset (MVSEC) [176] which is available online [177]. This is the only large publicly available dataset acquired with a real event-based stereo system, and over recent years it has become the standard [154, 160] for evaluation of event-based stereo methods. It is collected by a system composed of a LIDAR and two event-based cameras with a resolution of 346 × 260 pixels mounted on various vehicles, such as a drone, a car, and a motorcycle. The LIDAR acquires frames with sparse depth measurements at 20Hz, while the event-based cameras acquire continuous streams of events and gray-scale video frames, which we use for visualization purposes only.

We unpack the original data in the ROS bag format [178] to a more convenient format for training. The depths we convert to left-view sub-pixel disparities and save as images. We preserve sub-pixel precision by scaling the disparities. We assume that all pixels with disparities > 36 have unknown disparities. Then, for each depth, we find the closest gray-scale image in time, and events preceding the depth by 0.05 seconds in the left and right view. We correct their optical distortions, rectify and save them.

In this work, we use the *Indoor Flying* dataset from MVSEC, which is captured from a drone flying in a room with various objects. Similar to [154], we compute and report the *mean-absolute-depth-error* (MDE) and the *one-pixel-error* (1PE), computed in sparse locations corresponding to 15’000 events preceding each depth measurement. We report the results for three data splits summarized in Table 5.4. Following the [154], we exclude the take-off and landing frames from the dataset. For testing, we use frames from the same intervals as [154] to guarantee consistent and comparable results.

5.4.2 Importance of spatial and temporal context

In this section, we analyze the importance of temporal and spatial context for event-based stereo. The amount of the spatial context is determined by the receptive field of the stereo network with which we combine a temporal aggregation module. The larger receptive field allows the network to access a larger spatial context. The amount of the temporal context, available to the network, we can control by setting the event queue capacity.

To understand the importance of the temporal context, we firstly combine the proposed temporal aggregation with the small receptive field stereo network (MC-CNN [19]) and then with the large receptive field stereo network (PDS [173]). In each case, we train the resulting network two times using different random initializations on the first split of the Indoor Flying dataset, varying the event queue capacity. To avoid over-fitting, in each trial we select the network that achieves the lowest error on the validation set over the training epochs. In Table 5.5 we show mean depth errors on the validation set averaged across the trials. In all experiments, we set the time horizon of the event queue to $\tau = 0.5$ seconds.

From the table, it is clear that when the large spatial context is not available, the larger temporal

Table 5.4 – Summary of the Indoor Flying dataset splits. For each split, we specify which sequences and frames we use for training, validation and test. For example, $S^1_{140...1200}$ means that we use frames from 140 to 1200 exclusively from sequence one. Note that during the test we use frames from the same intervals as [154] in order to guarantee consistency of the test results.

Split #	**Set**	**Sequence and frames**	**Size**
1	Training	$S^2_{160...1580} \cup S^3_{125...1815}$	3110
	Validation	$A \in S^1_{140...1200}$	200
	Test	$B \in S^1_{140...1200} \mid A \cap B = \emptyset$	861
2	Training	$S^1_{80...1260} \cup S^3_{125...1815}$	2870
	Validation	$A \in S^2_{120...1420}$	200
	Test	$B \in S^2_{120...1420} \mid A \cap B = \emptyset$	1101
3	Training	$S^1_{80...1260} \cup S^2_{160...1580}$	2600
	Validation	$A \in S^3_{73...1615}$	200
	Test	$B \in S^3_{73...1615} \mid A \cap B = \emptyset$	1343

Table 5.5 – Importance of temporal context in the presence of large and small spatial context. The numbers in the table show mean depth errors on the validation set of the first split of the Indoor Flying dataset, averaged over two trials. Notice, that when the large spatial context is not available, the larger temporal context helps to achieve the lower error. However, when the large spatial context is available, the larger temporal context is ignored.

	Queue capacity κ, [events]			
	1	**3**	**7**	**15**
Small spatial context, MC-CNN [19]	80.4	71.5	67.7	59.4
Large spatial context, PDS [173]	13.3	13.4	13.5	13.3

context helps to achieve the lower error. However, when the large spatial context is available, the larger temporal context is ignored, probably because it is less reliable. These results, however, are valid only for the given highly-dynamic stereo dataset, filmed from the drone, and the stereo matching problem. We suspect that for other datasets and problems, such as monocular depth estimation or optic flow, a large temporal context provided by the proposed temporal aggregation approach can be very useful, even for networks with a large receptive field.

5.4.3 Effectiveness of the temporal aggregation

In this section, we compare the proposed temporal aggregation to the hand-crafted baseline in combination with the small receptive field (MC-CNN [19]) and the large receptive field (PDS [173]) stereo networks. In each case, we train the network two times using different

Table 5.6 – Effectiveness of the proposed temporal aggregation method compared to the baseline. We report test set results on the first split of the Indoor Flying dataset averaged over two trials. Our method outperforms the baseline with the MC-CNN network with a small 9 × 9 receptive field because it allows utilizing large temporal context in form of timestamps of multiple individual events, which very helpful when the network is deprived of a large spatial context. However, when the proposed temporal aggregation is combined with PDS [173] stereo network with a large receptive field, its advantage compared to the baseline is not large.

	Mean depth error, [cm]	**1-pixel-error, [%]**
	Small spatial context (MC-CNN [19])	
Hand-crafted	94.5	35.9
Continuous fully-connected	58.4	30.5
	Large spatial context (PDS [173])	
Hand-crafted	13.3	8.9
Continuous fully-connected	12.7	8.0

random initializations on the first split of the Indoor Flying dataset. To avoid over-fitting, in each trial we select the network that achieves the lowest error on the validation set over all epochs. In Table 5.6 we show mean depth errors on the test set averaged across two trials.

As shown in the table, the proposed temporal aggregation outperform the hand-crafted method, when it is combined with the MC-CNN network with a small 9 × 9 receptive field. This is because it utilizes a large temporal context in the form of precise timestamps of multiple individual events, which is very helpful when the network is deprived of a large spatial context. However, when the proposed temporal aggregation is combined with PDS [173] stereo network with a large receptive field, its advantage compared to the baseline is not large. Note, however, that in contrast to the baseline, our method does not require parameter search since it discovers appropriate amount of temporal context during training, as we showed in the previous section.

For the stereo matching, the PDS stereo network has better performance than MC-CNN, therefore, in the following sections, we combine the proposed temporal aggregation method with this network and refer to it as *Deep Dense Event Stereo* (DDES).

5.4.4 Benchmarking

In this section, we compare the proposed stereo method to the state-of-the-art event-based methods [154, 155, 160], published recently, and to two traditional methods [35, 58] operating on event images implemented in [160].

For quantitative comparison we use the protocol from [154] described in §5.4.1. According to this protocol, we evaluate results in sparse locations corresponding to 15’000 most recent events. We use the same parameters and experiment settings as in §5.4.3.

During our experiments with various splits of the Indoor Flying dataset, we noticed significant differences between the test and the training set of the second split. In the test set, composed of the second sequence, there are much more abrupt motions (triggering large numbers of events) than compared to the training set, composed of sequences one and three. This suggests that the test and the training set of this split are drawn from very different underlying distributions and thus should not be used in the experiments. As a partial remedy, for the second split, we train the network using a fixed number of 130’000 events instead of a fixed time horizon and show the results in the table.

The benchmarking results are summarized in Table 5.7. Our proposed DDES method significantly outperforms other single viewpoint methods, such as TSES [154], CopNet [155], SGM* [58, 160] and FCVF* [35, 160] and even performs on-par with Semi-Dense 3D method [160] which fuses depths from several viewpoints using known camera motion. We also train and test our method using full ground truth, taking into account all locations, including those without events. The results are only slightly worse than results with the sparse ground truth. To our knowledge, the proposed method is the only method capable of producing dense stereo results for event-based cameras.

For qualitative comparison, we estimate disparity using DDES trained on the full ground truth for example cases similar to the ones used in [154, 160]. Figure 5.8 contains a visual comparison of our results with those of TSES [154] and Semi-Dense 3D [160] borrowed from the respective papers. Unlike previous methods, the proposed method computes truly dense and sub-pixel accurate disparity.

Our proposed method runs at about 10 frames per second on a desktop PC with a GeForce GTX TITAN X GPU.

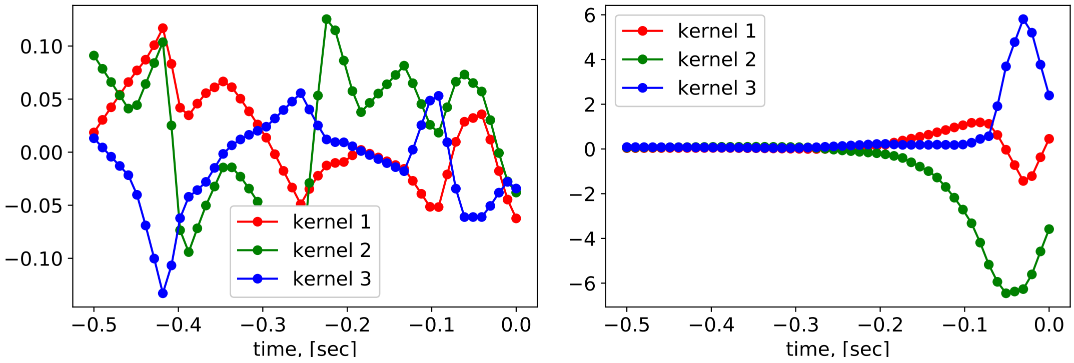
5.4.5 What does continuous fully-connected layer learn?

In this section, we visualize the output of the kernel network. To this end, we input uniformly sampled timestamps $\in [-0.5, 0]$ to the kernel network and plot every row of the CFC weights tensor as a smooth curve, which we call *weight kernel*.

We show resulting kernels before and after the training in Figure 5.7. At the beginning of the training, the weight kernels produced by the kernel network are normally distributed, thanks to the custom initialization. After the training, the weight kernels become smooth in time and converge to one of two shapes: bell-shaped (kernels 2 and 3) or derivative (kernel 1). The bell-shaped kernels detect events with particular timestamps, while the derivative kernels compute event count changes (time-derivative) at varying time scales. Most of the kernels assign close to zero weights to old events.

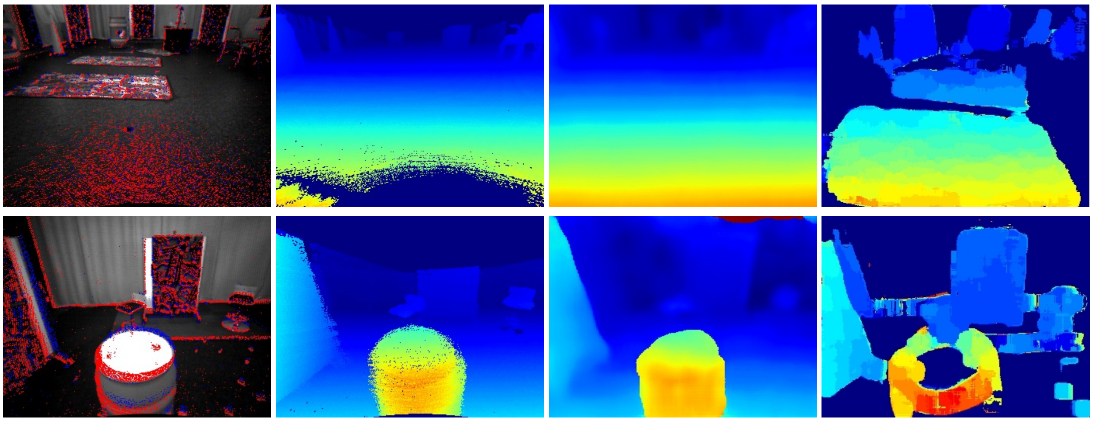
Table 5.7 – Benchmarking on the Indoor Flying dataset using sparse ground truth, corresponding to the last 15'000 events, and dense ground truth. Results for TSES [154] and CopNet [155] are from [154] and results for Semi-Dense 3D [160], SGM* [58, 160] and FCVF* [35, 160] are from [160]. SGM* and FCVF* methods implemented in [160] are similar to the original frame-based methods but operate on event images. For Semi-Dense 3D, SGM* and FCVF* results for the second split are not available. We report average test set errors over the three randomized training trials. All methods are sorted in ascending order according to their test error. Our proposed method dubbed Deep Dense Event Stereo (DDES) is highlighted. Note, that it outperforms other single viewpoint methods, such as TSES, CopNet, SGM* and FCVF*, and even performs on-par with Semi-Dense 3D method that fuses depths from several viewpoints using known camera motion. The proposed method performs only slightly worse when trained and tested on the dense ground truth. Notably, the proposed method is the only method capable of producing dense results.

Method	**Mean absolute depth error, [cm]**		
	Split 1	**Split 2**	**Split 3**
Sparse ground truth (last 15'000 events)			
Semi-Dense 3D [160]	13	–	33
DDES (proposed)	13	18	18
TSES [154]	36	44	36
CopNet [155]	61	100	64
SGM* [58, 160]	93	–	119
FCVF* [35, 160]	99	–	103
Dense ground truth			
DDES (proposed)	17	29	28

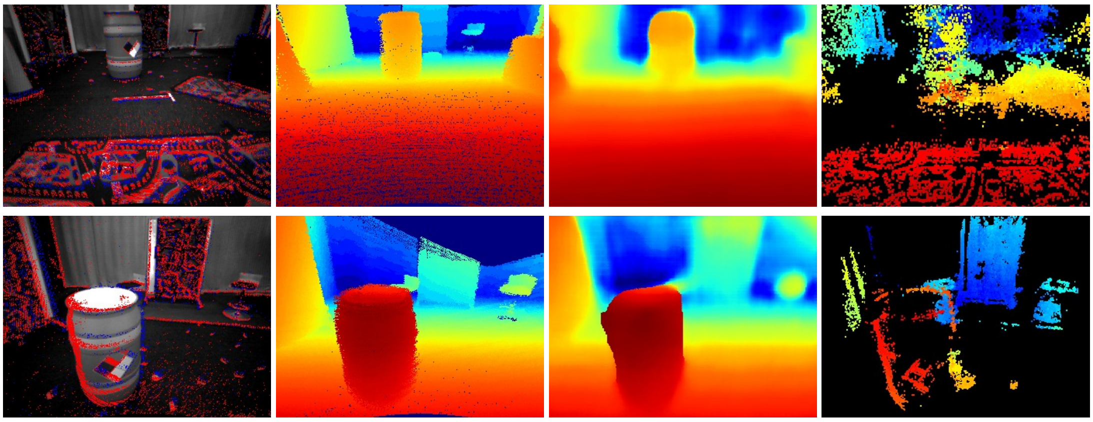


(a) Before training (b) After training

Figure 5.7 – Visualization of kernel network output before (a) and after (b) the training. Before the training (a), the weight kernels produced by the kernel network are normally distributed by design. After the training (b), the weight kernels have one of two shapes: bell-shaped (kernels 2 and 3) and derivative (kernel 1).



(a) Events (b) Ground Truth (c) DDES (proposed) (d) TSES [154]



(a) Events (b) Ground Truth (c) DDES (proposed) (f) Semi-Dense 3D [160]

Figure 5.8 – Qualitative comparison with recent event-based methods on the Indoor Flying dataset. For comparison, we select frames similar to the ones used in [154] and in [160]. Results for TSES [154] and Semi-Dense 3D [160] are borrowed from the respective papers. Note, that, unlike our method, Semi-Dense 3D fuses depth from several viewpoints using known camera motion. The rows correspond to frame #100 from sequence 1, frame #340 from sequence 1, frame #1700 from sequence 3 and frame #980 from sequence 1 correspondingly. To get the results for sequence 1 we train the network using sequences 2 and 3 (first split) and to get the results for sequence 3 we train the network using sequences 1 and 2 (third split). We try to match the color-coding of the methods as much as possible. In all figures, the warmer colors correspond to the closer objects. In (a) we visualize the 15’000 most recent events from the left camera, overlaid with a gray-scale image, which is not used by the methods. Positive events are shown in red and negative events are shown in blue. In (b,c,d) locations without disparities are shown in dark blue and in (f) in black. Note, that our proposed method (c) computes dense disparities, while in TSES (d) some disparities are invalidated by outlier rejection and in Semi-Dense 3D (f) disparities are computed only for locations with events. Similarly to Semi-Dense 3D method, our proposed method (c) estimates sub-pixel disparities, while TSES (d) estimates integer disparities.

5.5 Conclusion

In this work, we propose a novel learning-based method for embedding event sequences recorded by event-based vision sensors. It explicitly treats events as a stream of sparse 3d data points, each with two discrete spatial coordinates and one continuous temporal coordinate, and takes into account timestamps and spatial positions of all events in a time interval. We show that when the network has small receptive field our proposed embedding outperforms state-of-the-art hand-crafted embedding in stereo matching task.

Using the proposed embedding we develop a deep neural network for stereo matching. This is the first learning-based stereo matching method for event-based cameras and the only method to date producing dense results. We show that the proposed method outperforms the latest state-of-the-art methods on the standard MVSEC dataset by large margins.

Event-based cameras offer advantages such as higher dynamic range and temporal resolution over traditional frame-based cameras. However, the special data structure of an event stream requires a new suite of imaging algorithms. We believe that the embedding that we develop in this work could serve as a building block also beyond stereo.

Applied part Part II

6 Calibration of a satellite stereo system

Contents

6.1 Introduction **78**

6.2 Background **79**

6.2.1 Camera model 81

6.2.2 Calibration using starfield images 82

6.2.3 Space image processing tools 83

6.3 Optical distortion model selection **83**

6.4 On-ground calibration **83**

6.5 In-flight calibration **85**

6.5.1 Method 85

6.5.2 Results 87

6.6 Color mosaicing **91**

6.6.1 Method 91

6.6.2 Results 92

6.7 Stereo reconstruction **94**

6.7.1 Method 94

6.7.2 Results 96

6.8 Conclusion **98**

This chapter is based on the following publication:

S. Tulyakov, A. Ivanov, N. Thomas, V. Roloff, A. Pommerol, G. Cremonese, T. Weigel and F. Fleuret, Geometric calibration of Colour and Stereo Surface Imaging System of ESA's Trace Gas Orbiter. *Advances in Space Research*, 61(1):487–496, 2018.

6.1 Introduction

On March 15, 2016, European Space Agency (ESA) launched Trace Gas Orbiter (TGO) satellite to Mars, as a part of the ExoMars project. Main aim of the satellite is to find trace gases - evidence of geological or biological activity on Mars. Color and Stereo Surface Imaging System (CaSSIS) on board of TGO helps to characterize sites identified as potential sources of trace gases visually and structurally. To this end, one can use color mosaics and Digital Terrain Models (DTMs), that can be produced from raw CaSSIS data. To produce these scientific products, however, we need precise geometric parameters of the telescope, such as its focal length, optical distortion model and orientation of the camera relative to the spacecraft that can be found using a *geometric calibration*. While there are many calibration methods for a conventional camera, we cannot use them for the calibration of a telescope with a large focal length and complex optical distortion, such as CaSSIS. We address this problem in our work.

Main contributions of this work can be summarized as following:

1. We propose a novel geometric calibration method for a satellite telescope with a large focal length and complex optical distortion that relies on starfield images and apply the method to refine CaSSIS parameters.

2. We develop tools for reconstruction of bundle-adjusted color mosaics and DTMs from raw CaSSIS data and experimentally show that color bands misalignment in the color mosaics and average absolute vertical error of the DTMs with respect to overlapping HiRISE DTMs are small.

This chapter is organized as follows

First, in the “Background” section we introduce CaSSIS, then, in § 6.2.1 present its geometric camera model, in § 6.2.2 review calibration methods that rely on images of starfields, and in § 6.2.3 overview tools for processing space images that we use in this work.

Next, in the “Optical distortion model selection” section we chose an optical distortion model for CaSSIS using optical simulation data. In the “On-ground calibration” section we briefly discuss the unsuccessful attempt to perform calibration of the CaSSIS in laboratory, using collimator and calibration target. Afterward, in the “In-flight calibration” section in § 6.5.1 we present novel calibration method that relies on starfield images, and in § 6.5.2 show calibration results, obtained using this method for the CaSSIS.

Then, in the “Color mosaicing” section, in § 6.6.1 we propose an automatic tool for reconstruction of color mosaics from raw CaSSIS data, and in § 6.6.2 perform qualitative and quantitative analysis of reconstructed color mosaics, focusing on a misalignment of color bands. In the following “Stereo reconstruction” section, in § 6.6.1 we similarly propose an automatic tool for DTM reconstruction from raw CaSSIS data, and in § 6.6.2 perform analysis of quality of the DTMs, using an overlapping DTMs from another stereo system with a higher resolution. Finally, in the “Conclusion” section we summarize our results.

6.2 Background

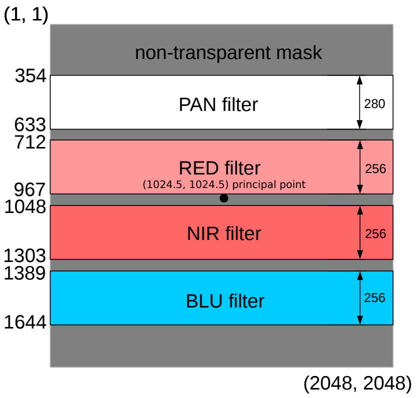
The CaSSIS ([179–183]) is a multi-spectral push-frame camera with 4 rectangular color filters covering its sensor. Its brief specification is provided in Figure 6.1. As the spacecraft is moving along an orbit, each part of a targeted area becomes visible, in each filter, and gets sequentially filmed by the CaSSIS as shown in Figure 6.2. We call individual images of this sequence *exposures* and sub-images acquired by individual filters *framelets*. On the ground, we can collate the framelets and stitch them using camera parameters and telemetry into a color mosaic of the target area.

The CaSSIS is also a stereo system. It is capable of acquiring two sequences from different viewpoints on the same orbit as shown in Figure 6.3. While approaching a target area it acquires the first sequence of exposures, then, it mechanically rotates and acquires the second sequence, while departing from the target area. By computing parallax between these two sequences, we can reconstruct a 3d model of surface relief in the target area, called *Digital Terrain Model (DTM)*. Compared to other color cameras with 3d reconstruction capabilities orbiting Mars, the CaSSIS has several unique features, such as high spatial resolution, second only to High-Resolution Imaging Science Experiment (HiRISE), and ability to acquire stereo pairs from the same orbit within a very short time interval.

To prepare scientific products, such as color mosaics and DTMs from raw CaSSIS data, we need precise geometric camera parameters, such as its focal length, optical distortion model and rotation relative to the spacecraft frame. While we know their nominal values from a technical specification, their actual values might deviate from the nominal ones, due to imprecise manufacturing, mounting, or various changes during the spacecraft cruise and operation due to structure dryout and zero gravity. Therefore, we have to measure their actual values in a controlled environment of a clean room and validate during a commissioning phase in flight. This is the main goal of a geometric calibration procedure.

There are many geometric calibration methods [22, 184–186] and tools [187–189] for conventional frame cameras. However, we can not use these off-the-shelf tools for the calibration of telescopes such as CaSSIS, for two reasons. Firstly, most of them require images of calibration targets, such as a checkerboard chart. For telescopes with a large focal length, however, such targets must be very large ($\approx km^2$) and should be placed very far away from the telescope ($\approx 10\ km$), which is impractical. Secondly, telescopes often have off-axis optical designs with complex optical distortion, that cannot be handled by off-the-shelf tools. Therefore, there is a need for specialized calibration methods, which are unfortunately scarce in the literature. In this work, we address this problem by proposing a universal method for calibration of a telescope with a large focal length and complex optical distortion and apply this method to the CaSSIS.

After refining the geometric parameters of the CaSSIS, we produce scientific products from raw data. Unfortunately, there is no out-of-box universal tool capable of doing this. Therefore, we develop fully automatic tools for the color mosaicing and DTM reconstruction for the



(a) Sensor outline

Parameter	Specification
Optic	3-mirror plus fold mirror off-axis
Detector	Raytheon Osprey 2k CMOS hybrid
Filters	675, 485, 840, 985 nanometers
Focal length	880 millimeters
F#	6.52
Pixel size	10 × 10 micrometers
Detector area	2048 × 2048 (2048 × 1350 is used)
FOV	1.33 × 0.88 degree
Angle w.r.t nadir	10.0 ± 0.2 degree

(b) Specifications

Figure 6.1 – CaSSIS sensor outline (a) and specifications (b). The CaSSIS telescope is a three-mirror anastigmat system (off-axis) with a fold mirror. The CaSSIS Filter Strip Assembly (FSA) comprises a Raytheon Osprey 2048×2048 hybrid CMOS detector with 4 colour filters mounted on it following the push-frame technique. Narrow dark bands between the filters reduce spectral cross-talk. The detector can acquire an un-smeared image along the ground-track.

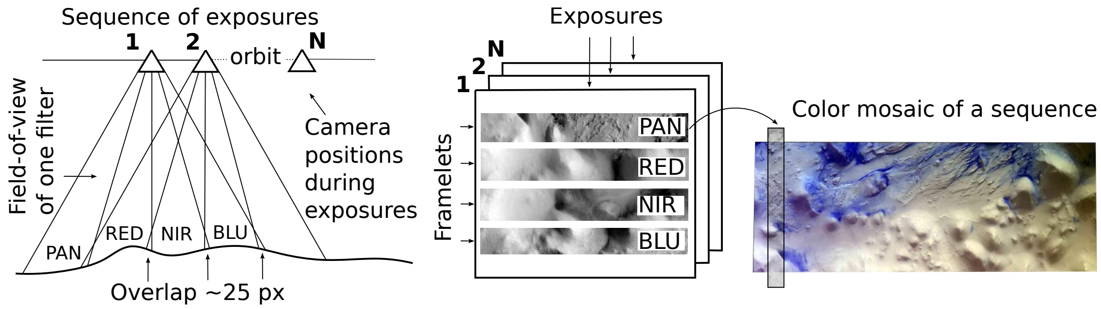


Figure 6.2 – Color image acquisition capability of the CaSSIS. The CaSSIS acquires a *sequence* of images of a target area from different locations as it moves along an orbit. We call individual images *exposures* and sub-images acquired by individual filters – *framelets*. On the ground, we can stitch framelets into a color mosaic of the target area using camera parameters and telemetry.

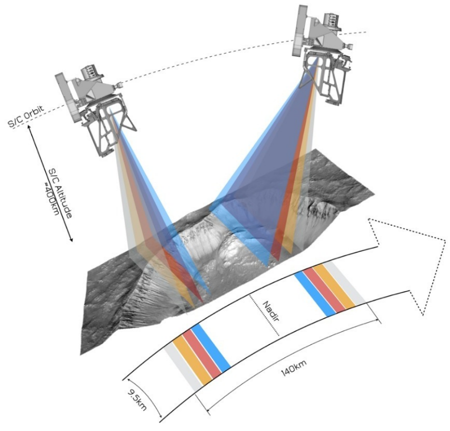


Figure 6.3 – Stereo acquisition capability of the CaSSIS. While approaching a target area the CaSSIS acquires the first sequence of exposures, then it mechanically rotates and acquires the second sequence, while departing from the target area. By computing parallax between these two sequences, we can reconstruct a 3d model of surface relief in the target area, called *Digital Terrain Model (DTM)*. Courtesy of European Space Agency (ESA).

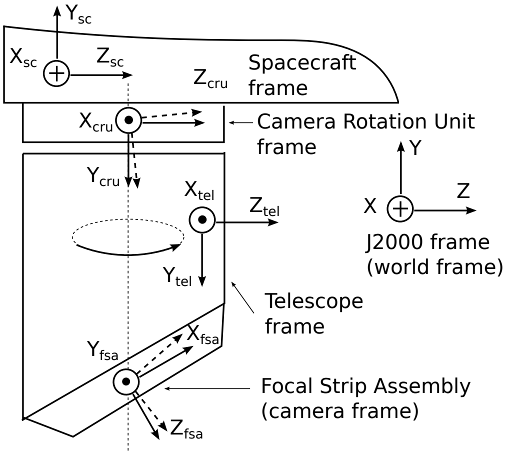


Figure 6.4 – Visualization of CaSSIS coordinate frames. We set the world frame equal to *Equatorial J2000 (J2000)* frame and the camera frame we call *Focal Strip Assembly (FSA)* frame. We know the rotation of the *Spacecraft (SC)* frame with respect to the J2000 thanks to spacecraft attitude sensors, such as star trackers, and the rotation of the *Telescope (TEL)* frame with respect to the *Camera Rotation Unit (CRU)* frame because we control the rotation of the CaSSIS ourselves. Also, we know the nominal rotation of the Camera Rotation Unit frame relative to the Spacecraft frame and the rotation of the Focal Strip Assembly frame relative to the Telescope frame from the specification (dashed axis) but want to refine them during geometric calibration.

CaSSIS.

6.2.1 Camera model

The CaSSIS camera model is similar to the one described in Chapter 2, however, there are several important differences which we highlight in this section.

Extrinsic model. In the case of the CaSSIS, we set the world frame equal to *Equatorial J2000 (J2000)* frame, because in star catalogs star positions are given in this frame, and refer to the camera frame as *Focal Strip Assembly (FSA)* frame. Furthermore, when we work with images of starfields, a translation of the camera relative to world frame is negligible compare to a distance to the calibration targets (stars). Therefore, we set the translation vector **t** in Equation 2.1 equal to zero. The camera rotation matrix we factorize as

$$\mathbf{R} = \widehat{\mathbf{R}}_{TEL\rightarrow FSA} \cdot \mathbf{R}_{CRU\rightarrow TEL} \cdot \widehat{\mathbf{R}}_{SC\rightarrow CRU} \cdot \mathbf{R}_{J2000\rightarrow SC}, \tag{6.1}$$

where $\mathbf{R}_{J2000\rightarrow SC}$ is a rotation of the *Spacecraft (SC)* frame with respect to the *J2000 Equatorial frame*, $\mathbf{R}_{SC\rightarrow CRU}$ is a rotation of the *Camera Rotation Unit (CRU)* frame with respect to the Spacecraft frame, $\mathbf{R}_{CRU\rightarrow TEL}$ is a rotation of the *Telescope (TEL)* frame with respect to the Camera Rotation Unit frame and $\mathbf{R}_{TEL\rightarrow FSA}$ is a rotation of the *Focal Strip Assembly (FSA)* frame with respect to the Telescope frame. We know the $\mathbf{R}_{J2000\rightarrow SC}$ matrix, thanks to spacecraft attitude sensors, such as star trackers, and the $\mathbf{R}_{TEL\rightarrow CRU}$ matrix, because we control the rotation of the CaSSIS ourselves. We also know the nominal values of the $\widehat{\mathbf{R}}_{TEL\rightarrow FSA}$ and the $\widehat{\mathbf{R}}_{SC\rightarrow CRU}$ from the camera specification but want to refine them during geometric calibration. We summarize all coordinate frames that we use in the factorization in Figure 6.4.

Intrinsic model. In the case of the CaSSIS, the only intrinsic parameter is its focal length f. The coordinates (x_0, y_0) of a principal point in Equation 2.2 we set equal to the center of the sensor, as shown in Figure 6.1 (a).

Optical distortion model. As we show in in § 6.3, the conventional optical distortion models, described in Chapter 2, cannot represent complex optical distortion in a camera with off-axis optical elements, such as the CaSSIS. Therefore, in this work we adopt *Rational distortion model* [190].

In the camera frame coordinates, we describe the Rational model as

$$\mathbf{X}_C = \mathbf{A}^{\text{corr}}\chi \text{ , where } \chi = \begin{bmatrix} X_{Cd}^2 & X_{Cd}Y_{Cd} & Y_{Cd}^2 & X_{Cd} & Y_{Cd} & 1 \end{bmatrix}^T, \tag{6.2}$$

where $\mathbf{A}^{\text{corr}}$ is a 3×6 *Rational distortion correction matrix.*

The model has several interesting properties. Firstly, although in total it has 18 parameters, there are only 17 free parameters, since we can multiply both sides of the equation by the same number. Secondly, there is no close-form analytical solution for the inverse model. In other words, given ideal coordinates, we can not compute the corresponding distorted coordinates analytically. However, the Rational model can very precisely represent the inverse of itself [191]. We use this property and simultaneously estimate two Rational models: one that transforms the distorted coordinates to the ideal and characterized by $\mathbf{A}^{\text{corr}}$, and another that transforms the ideal coordinates to the distorted and characterized by $\mathbf{A}^{\text{dist}}$. Thirdly, we can estimate the parameters of the Rational distortion model from a single image of a calibration chart up to unknown homography without knowing the intrinsic and the extrinsic parameters of a camera as shown in [190].

6.2.2 Calibration using starfield images

As described in § 2.1.4, for geometric calibration of a camera we need images of calibration targets – objects with known real-world coordinates. Since precise angular positions of stars are documented in star catalogs [192], such as Two Micron All-Sky Survey (2MASS) and Tycho-2, starfields can serve as perfect calibration targets. Indeed, star trackers, that are an integral part of every spacecraft, often rely on starfield calibration [193–195]. Also, in [196] authors use starfields to calibrate consumer-level camera. Unfortunately, all known starfield-based calibration methods assume a simplistic optical distortion model, and therefore, we cannot use them for the calibration of a telescope with complex optical distortion.

Before stars from an image can be used for the calibration, they should be identified with stars from a star catalog. Fortunately, we can do this automatically [197] using the *Astrometry.net* library [198].

6.2.3 Space image processing tools

In this work, we rely on several well-established libraries for processing space images and computing DTMs. For preprocessing images, we use open-source *SPICE* [199] and *Integrated Software for Imagers and Spectrometers (ISIS)* [200] libraries. SPICE library helps to access spacecraft telemetry and characteristics, stored in a form of *SPICE kernels* [201], and allows performing various geometrical computations. This is a low-level library and we use it mostly in our in-flight calibration method. In contrast, USGS ISIS library provides high-level functions for image manipulation, that we use for the mosaicing and stereo triangulation. For the stereo matching, we rely on open-source *NASA Ames Stereo Pipeline (ASP)* [7, 202]. Besides the ASP, there are several alternatives, such as, for example, *SOCET SET* [203] and *S2P* [204, 205].

6.3 Optical distortion model selection

To find out what distortion model better represents CaSSIS optical distortion, we fit the Radial [22], the Brown-Conrady [23], the Rational [190] and the Bi-cubic [206] optical distortion models to the optical distortion data computed using a ray-tracing simulation provided by the telescope manufacturer (RUAG Space Zurich, Switzerland), and compare the average Euclidean errors of the models using a leave-one-out cross-validation.

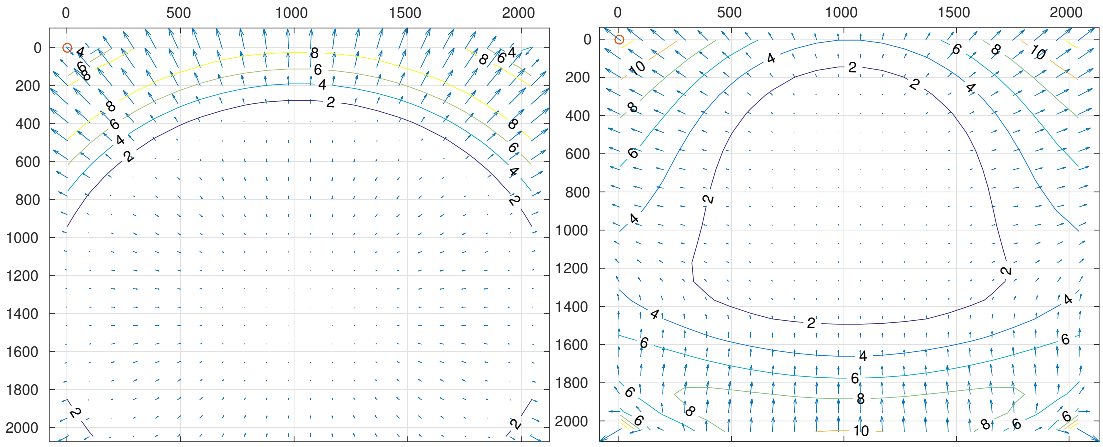
We show the resulting distortion fields and errors for each model in Figure 6.5. The Radial and Brown-Conrady models suffer from more than 1 pixel errors, and hence fail to represent the CaSSIS distortion, while the Bi-cubic and Rational models, with less than 0.1 pixel error, perform well. Among the last two models, we select the Rational model because we can easily estimate its parameters from a single image of a calibration chart and because it has fewer parameters than the Bi-cubic model.

6.4 On-ground calibration

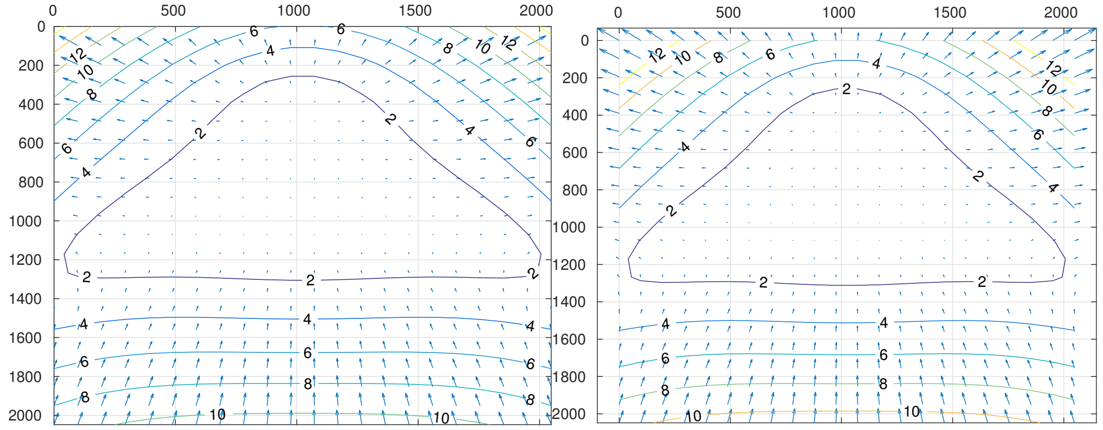
During our calibration experiment in a laboratory, we attempt to estimate the rational distortion model from a single image of a calibration chart, as in [190]. Because the focal length of CaSSIS is too large to acquire a sharp image of a calibration chart from a reasonable distance, we use the set-up with a collimator shown in Figure 6.7.

To identify dots in the acquired image, we apply adaptive thresholding and connected components detection methods. Then, we find the dot centers using centroid algorithm. Finally, we fit the regular rectangular grid to the dot centers, using a simple algorithm that starts from an arbitrarily-selected dot and expands the grid in horizontal and vertical directions, until no new dots can be added to the grid.

We show the acquired image with the fitted grid in Figure 6.6. Analysis of the grid confirms the presence of small optical distortion in the image: the grid rows and columns appeared,



(a) Radial ($\overline{\text{error}}$=3.169 pix) (b) Brown-conrandy ($\overline{\text{error}}$=1.585 pix)



(c) Rational ($\overline{\text{error}}$=0.088 pix) (d) Bi-cubic ($\overline{\text{error}}$=0.015 pix)

Figure 6.5 – Distortion fields of various optical distortion models fitted to the simulated CaSSIS optical distortion data computed using a ray-tracing. The vectors show a transformation from distorted to ideal image coordinates. The contour lines show a magnitude of this transformation. The errors are average Euclidean distances between positions of ideal pixels, as predicted by the model, and their actual positions. Note that the simple Radial (a) and the Brown-Conrady (b) models, with > 1 pixel error, fail to represent the CaSSIS distortion, while the Bi-cubic (c) and the Rational (d) models, with < 0.1 pixels error, both perform well.

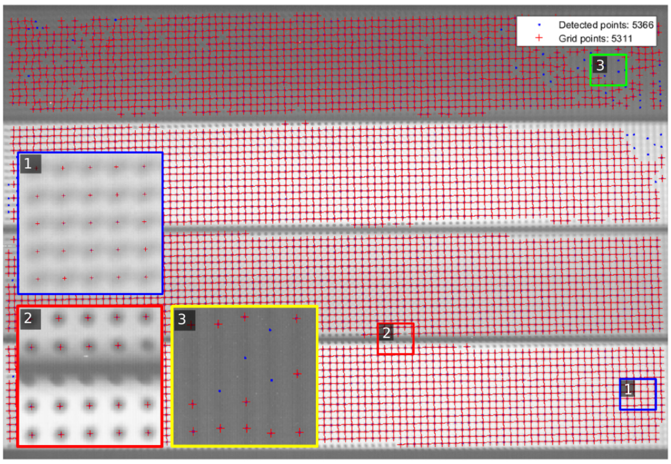


Figure 6.6 – Image of the calibration chart overlaid with the grid (blue filter is on the top). The red crosses show dots that are added to the grid and the blue points show ignored dots.

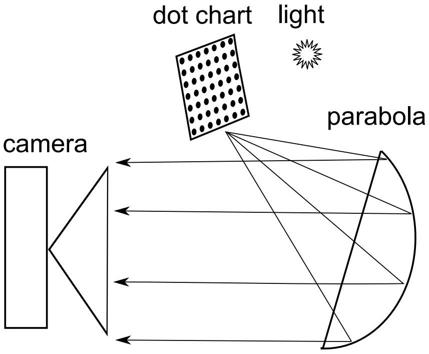


Figure 6.7 – On-ground calibration setting. To acquire a sharp image of the dotted calibration target from a reasonable distance, we put it in the focus of the parabolic collimator.

not as straight lines but, as high-order curves. However, we fail to estimate the distortion field resembling Figure 6.5 (c) using the grid. This is probably because the experimental data is contaminated with residual distortion coming from the off-axis collimator which we can not decouple from the CaSSIS distortion.

6.5 In-flight calibration

During TGO commissioning and mid-cruise checkout, the CaSSIS acquired multiple images of starfields, that we use for in-flight calibration. In this section, in § 6.5.1 we propose a novel calibration method that uses starfield images and in § 6.5.2 apply this method to refine the geometric parameters of the CaSSIS.

The proposed calibration method is implemented in Matlab using SPICE [199] and Astrometry [198] libraries. To encourage re-use of the method we make our code available on-line[1].

6.5.1 Method

We show the workflow of the proposed in-flight calibration method in Figure 6.8 and describe its steps below:

Exposures preprocessing. First, we assemble full-sensor images from several data packets according to information in XML files from the telemetry conversion. We denoise every image by subtracting the median of several close in time images from each image. This procedure helps to get rid of fixed-pattern noise and hot pixels. Then, we flatten each image by applying

---

[1] https://github.com/tlkvstepan/CaSSIS_calibration

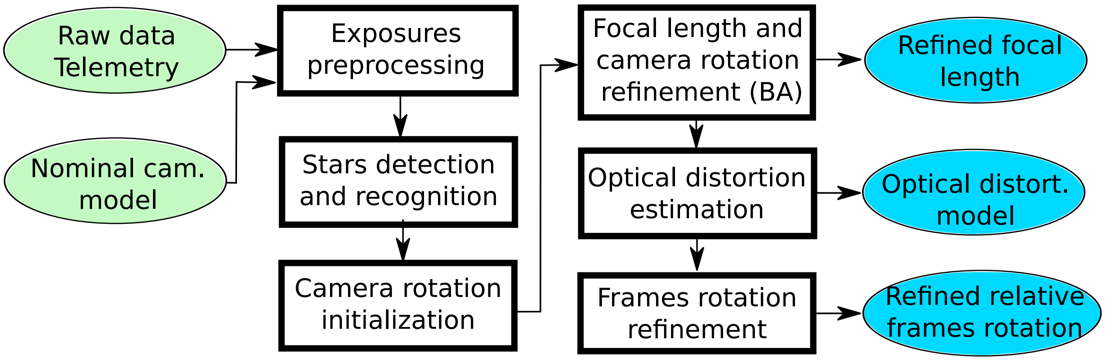

Raw data
Telemetry
Nominal cam. model
Exposures preprocessing
Stars detection and recognition
Camera rotation initialization
Focal length and camera rotation refinement (BA)
Optical distortion estimation
Frames rotation refinement
Refined focal length
Optical distort. model
Refined relative frames rotation

Figure 6.8 – Workflow of the in-flight calibration. The green ellipses show the input data, the blue ellipses show output data and the rectangles show processing steps.

a Difference-of-Gaussian (DoG) filter.

Starfields recognition. Next, we perform stars detection and recognition using the open-source Astrometry.net [198] library and Two Micron All-Sky Survey star catalog (2MASS) [192]. The library takes an image of a starfield as an input and outputs (x, y) coordinates of stars in the image, and their corresponding right ascension α and declination δ coordinates in the Equatorial J2000 frame (and epoch). In the next step, we collect information about detected stars from all images and filter out erroneous detections. First, we ignore detections with very low relative image brightness. Next, since the calibration image sets consist of sequences of 3-4 almost identical images, we mark a star as a false detection, if it is not re-detected at a similar position in at least 2 other images. Finally, to ensure that we have enough stars for a robust estimation of extrinsic parameters for every exposure, we remove exposures that have less than 10 stars.

Camera rotation initialization. We estimate the camera rotation for every image independently. During the estimation, we set the focal length of the camera to the nominal and search for the camera rotation that minimizes the *reprojection error*, *i.e* the Euclidean distance between observed and predicted star positions in each image individually (please, refer Chapter 2). We preform the optimization using Levenberg-Marquardt algorithm (lsqnonlin in MATLAB). We initialize the optimization with nominal rotation angles, which we read from the SPICE kernels of the ExoMars mission [201], using the SPICE Toolkit [199].

Iterative Bundle Adjustment (BA). In this step, we search for a refined focal length and rotations (extrinsic parameters) that minimize the reprojection errors for all images simultaneously. The optimization is performed with Levenberg–Marquardt algorithm. We initialize the optimization with the nominal focal length and the rotation matrices that we found in the previous step. After each BA iteration, we rejected as outliers stars that have much larger residual reprojection errors than their spatial neighbors and repeat the BA until no new outliers are found. Without this outlier rejection, the subsequent optical distortion estimation would fail.

Optical distortion estimation. In this step, we “freeze” the intrinsic and the extrinsic camera

Table 6.1 – Summary of the starfield datasets that we use in this work. Each dataset consists of several imaging experiments and each experiment consist of exposure sequences. Each sequence consists of 3-4 almost identical exposures, acquired within a short time interval. In each exposure, we detect 10-60 stars. In the table, we show number of the detected stars that pass the false detection removal procedure.

Set	Acquisition date	Experiment	# of exposures	# of stars
Training	04/13/2016	pointing cassis	197	3528
	06/14/2016	mcc motor		
	20/11/2016	stellar cal orbit09		
	24/11/2016	stellar cal orbit10		
Validation	04/07/2016	commissioning 2	8	598

parameters and search for a Rational optical distortion model that minimizes the remaining reprojection error. The optimization is performed with Levenberg–Marquardt algorithm. We initialize the optimization process using a “no distortion” hypothesis.

Frames rotation refinement. Finally, we search for the $\widehat{\mathbf{R}}_{SC\rightarrow CRU}$ and the $\widehat{\mathbf{R}}_{TEL\rightarrow FSA}$ matrices, introduced in § 6.2.1, that minimize an average angular error between the predicted camera pointing and camera pointing estimated from the image during the BA. The optimization is performed using Levenberg–Marquardt algorithm with *RANSAC* [22, p117-121] outlier rejection method. We initialize the optimization process using the nominal rotation matrices.

6.5.2 Results

In this section, we apply the proposed geometric calibration method to the CaSSIS. We estimate the camera parameters using the training set and check the results using a separate validation set. Both sets are summarized in Table 6.1. They consist of 5 starfield imaging experiments. We select these experiments since they contain images of dense starfields acquired with a long exposure. In these experiments, we use only sequences with exposure of at least one second. Furthermore, in each sequence, we ignore the first exposure, which has persistent brightness anomaly. In the training set, we identify 3528 stars that pass the false detection removal. As we show in Figure 6.9, these stars cover the sensor densely and uniformly, allowing for a good geometric calibration.

Using the stars detected in the training set, we refine the nominal camera rotation obtained from the SPICE kernel for every exposure, while keeping the focal length of the camera fixed to the nominal. By refining the rotations, we reduce the average reprojection error from 1219.4 (median 279.1) to 8.2 pixels. Notice, that the initial mean error is very large due to several anomalies in rotation sensor readouts.

Then, we use the estimated camera rotations and the nominal focal length to initialize the iterative Bundle Adjustment (BA) that refines the camera rotations and the focal length using all images simultaneously, while ignoring optical distortion. The BA converges after eleven

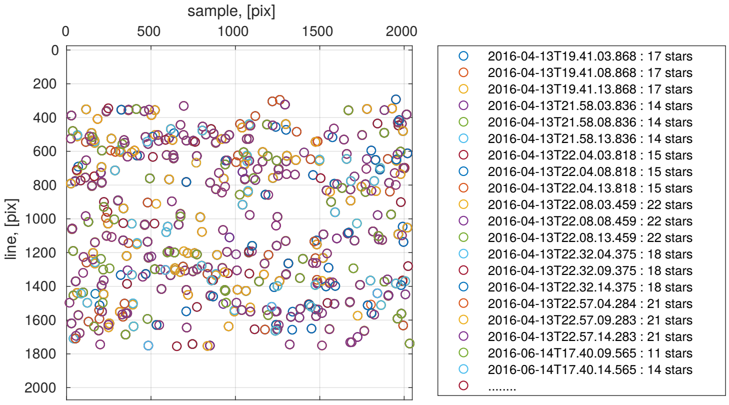


Figure 6.9 – Position of stars detected in the training set on the sensor. The stars cover the sensor densely and uniformly, allowing for a good geometric calibration. On the top and the bottom parts of the sensor, there are no observations, since they are covered by a non-transparent mask. There are in total 3528 stars.

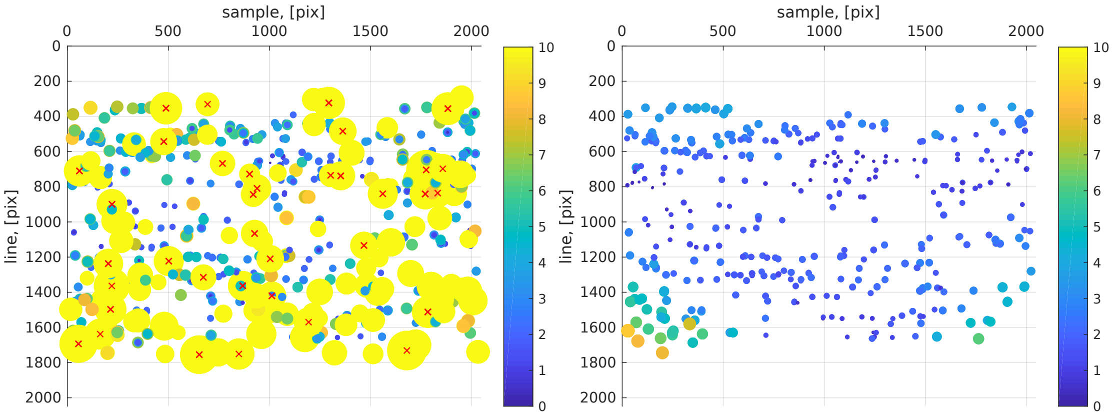
iterations. We show the effect of the iterative outliers rejection scheme in Figure 6.10. Note that after the first iteration, the BA residuals contain gross outliers, while after the last iteration, the residuals form a clear spatial pattern suggesting the presence of optical distortion. The BA reduces the reprojection error from 8.2 to 2.1 pixels. The focal length refined by the BA is equal to 874.9 millimeters, which is slightly shorter than the nominal focal length of 880 millimeters.

Then, we "freeze" the focal length and camera rotations and estimate the Rational distortion model. We show the estimated distortion field in Figure 6.11 (a). Note that its shape resembles the distortion field that we obtained in §6.3 by fitting the Rational model to optical simulation data, which we duplicate for convenience in Figure 6.11 (b), with an apparent 180 degrees rotation of the field. The rotation is probably caused by a particular sensor mounting relative to the lens.

We show the estimated parameters of the distortion correction model in Table 6.2. By taking into account the distortion model, we reduce the average reprojection error from 2.1 to 0.6 pixels. Moreover, as shown in Figure 6.12, after the distortion correction the residual reprojection error becomes spatially uniform and small when compared to the residual error after the bundle adjustment from Figure 6.10 (b). The apparent absence of a spatial pattern in the residuals suggests that they are caused by spatially uniform error, such as, for example, stars coordinate estimation error.

Next, we refine the rotation matrices $\widehat{\mathbf{R}}_{SC\to CRU}$ and $\widehat{\mathbf{R}}_{TEL\to FSA}$ and show the results in Table 6.4. During the refinement, we identify 60% of images as inliers and reduce the average angular error on these images from 0.272 to 0.045 degrees.

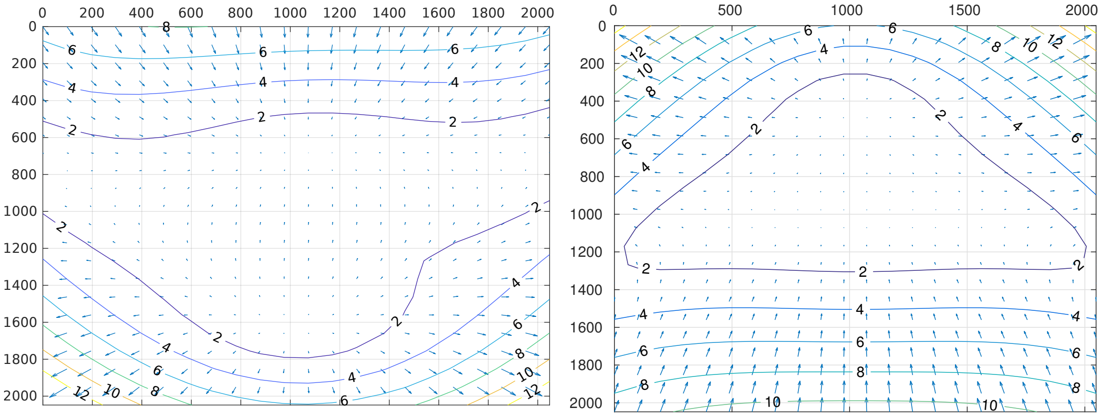
Finally, we compute errors of the estimated camera model on the validation set, that we did not use for the model estimation. Before the validation, we identify inlier star images and estimate image-based extrinsic parameters for the validation set using the same procedure as we used earlier during the camera parameters estimation. We summarize the validation results in Table 6.3. With the ideal extrinsic parameters, the refined focal length and the estimated optical distortion model reduce the reprojection error 7× from 3.5 to 0.5 pixels. However,



(a) First BA iteration

(b) Eleventh (final) BA iteration

Figure 6.10 – Residual reprojection error after the first and the eleventh BA iterations in the sensor coordinate frame. The color coding shows the actual scale of the residual error. The crossed-out residuals correspond to the identified outliers. On the top and the bottom parts of the sensor, there are no observations since they are covered by a nontransparent mask. Note, that after the first iteration (a), the residuals contain gross outliers, while after the eleventh iteration (b) the residuals form a clear spatial pattern suggesting the presence of optical distortion. The average reprojection error before the BA is 8.5 pixels, after the first iteration it is 7.0 pixels, and after the eleventh iteration, it is 2.1 pixels.



(a) Estimated from starfield images

(b) Estimated from simulation data

Figure 6.11 – Comparison of the distortion fields estimated from starfield images (a) and the optical simulation data (b), as described in §6.3. The vectors show the transformation from distorted to ideal image coordinates. The contours show the magnitude of the transformation. Note that the distortion fields are very similar in shape, with an apparent 180 degrees rotation of the field. The rotation is probably caused by a particular sensor mounting relative to the lens.

Table 6.2 – Parameters of the Rational distortion correction model estimated using starfield images. $\mathbf{A}_1^{\text{corr}}$, $\mathbf{A}_1^{\text{corr}}$ and $\mathbf{A}_3^{\text{corr}}$ are rows of the Rational distortion correction matrix.

$\mathbf{A}_1^{\text{corr}}$	0.0054	0.0024	0.0000	0.9994	-0.0001	0.0016
$\mathbf{A}_2^{\text{corr}}$	0.0001	0.0054	0.0025	-0.0004	0.9972	-0.0177
$\mathbf{A}_3^{\text{corr}}$	0.0000	0.0000	0.0000	0.0054	0.0016	1.0000

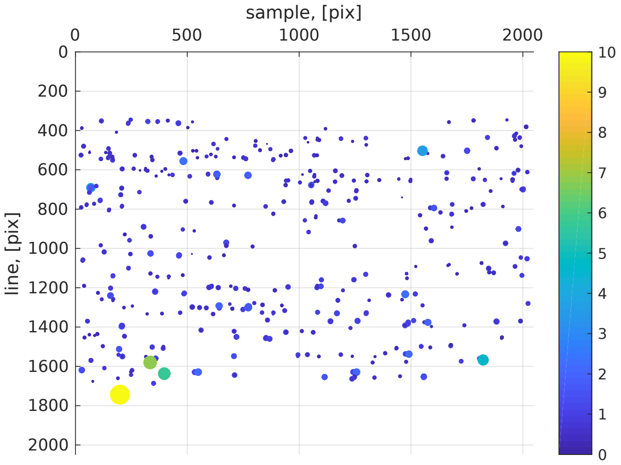


Figure 6.12 – Residual reprojection error after optical distortion correction. The average reprojection error is 0.6 pixels. The color coding shows an actual error scale, which is similar to Figure 6.10 (b). Note that the residuals are smaller and more spatially uniform than compared to the residuals after the BA from Figure 6.10 (b). The apparent absence of a spatial pattern in the residuals suggests that they are caused by some spatially uniform error, such as, for example, stars coordinates error.

Table 6.3 – Reprojection error on the validation set, which we did not use for model estimation. With the ideal extrinsic parameters, the refined focal length and the estimated optical distortion model reduces the reprojection error 7× from 3.5 to 0.5 pixels. However, with the refined rotation matrices the results are significantly worse that with the ideal. Perhaps, this is due to a non-systematic pointing error, that cannot be corrected by the calibration.

Frames rotation	**Focal length**	**Optical distortion**	**Error, [pix]**
nominal	nominal	no	275.9
ideal	nominal	no	3.5
	refined	no	1.9
	refined	estimated	0.5
refined	refined	estimated	8.3

Table 6.4 – Rotation matrices $\widehat{\mathbf{R}}_{SC\to CRU}$ and $\widehat{\mathbf{R}}_{TEL\to FSA}$, described in § 6.2.1, refined using starfield images. We represent every rotation matrix by three Euler angles, each specified by the rotation axis and counter-clockwise rotation angle in degrees. For example, $\alpha_X = 0.4$ denotes counter-clockwise rotation around X-axis by 0.4 degrees.

Rotation matrix	**Euler angles, [deg]**		
$\widehat{\mathbf{R}}_{SC\to CRU}$ (refined)	$\alpha_X = 0.021$	$\alpha_Y = 0.120$	$\alpha_Z = -179.881$
$\mathbf{R}_{SC\to CRU}$ (nominal)	$\alpha_X = 0.000$	$\alpha_Y = 0.000$	$\alpha_Z = -180.000$
$\widehat{\mathbf{R}}_{TEL\to FSA}$ (refined)	$\alpha_Y = 89.714$	$\alpha_X = 80.005$	$\alpha_Z = 0.168$
$\mathbf{R}_{TEL\to FSA}$ (nominal)	$\alpha_Z = 0.000$	$\alpha_Y = 90.000$	$\alpha_X = 80.110$

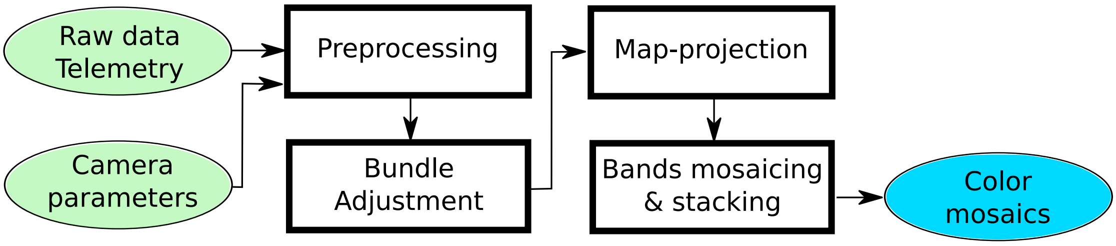


Figure 6.13 – Workflow of the proposed color mosaicing method.

with the refined rotation matrices the results are significantly worse than with the ideal. This suggests the presence of a non-systematic camera pointing error, for example, due to a camera jitter, that cannot be corrected by the geometric calibration.

6.6 Color mosaicing

In this section, in § 6.6.1 we propose a novel tool for reconstruction of color mosaics from raw CaSSIS data, and in § 6.6.2 provide a qualitative and quantitative analysis of their quality. We implement the tool in Python using USGS ISIS library [200] and make it available on-line[2].

6.6.1 Method

We show the workflow of the proposed mosaicing method in Figure 6.13 and describe each step below.

Preprocessing. First, we convert data packet for each framelet to ISIS ".cub" format and add telemetry information from SPICE kernels to each ".cub". We denoise BLU and NIR band framelets since they contain a lot of salt-and-pepper and Gaussian noise by applying median and Gaussian filters.

Bundle Adjustment (BA). In § 6.5.2 and § 6.6.2 we show that the CaSSIS is affected by small periodic high frequency motions, called *camera jitter*. This jitter cases small camera orientation errors that cannot be corrected by the geometric calibration and cause a framelets misalignment in a mosaic. To address this problem, we introduce into the pipeline BA that refines camera orientations for each exposure. For the BA, we use points matched across the bands, because an overlap between framelets of the same band is too small (≈ 25 pixels) to guarantee a reasonable amount of matches. During the BA, we penalize significant deviations from the nominal camera orientation to prevent divergence of BA and ensure that framelets from the same exposure share camera parameters. Since the camera jitter is relatively small, the BA works mostly in an automatic mode.

Map-projection. Next, we project all framelets from all bands to sinusoidal map, with opti-

---

[2] https://github.com/tlkvstepan/CaSSIS_color_and_stereo

Table 6.5 – Summary of the CaSSIS observations from “MTP07”, “STP027” and “boot2” that we use for the qualitative analysis of color mosaics.

Observation	Acquisition date and time	Center (lat, lon), [deg]	Resolution [m/px]
MY34_004060_181_1	21-10-2018 at 09:09:00	(0.125, 342.573)	4.6
MY34_004060_228_1	21-10-2018 at 09:24:28	(-44.847, 355.908)	4.4
MY34_004071_200_1	22-10-2018 at 06:52:20	(-18.457, 28.245)	4.4

Table 6.6 – Average along-track, across-track and Euclidean misalignment between the “NIR” and the “PAN” bands in 26 color mosaics from “MTP07”, “STP027” and “boot2” with standard deviations. The result obtained with the refined CaSSIS parameters and the BA, highlighted in the table, has the smallest band misalignment.

Settings	Along-track, [px]	Across-track, [px]	L^2, [px]
Nominal param.	2.7 ± 1.8	-1.7 ± 1.4	4.4 ± 0.6
Refined param.	0.8 ± 0.9	0.1 ± 0.8	1.8 ± 0.5
Refined param. & BA	0.4 ± 1.0	0.3 ± 0.4	1.3 ± 0.7

mally select parameters such as resolution and latitude-longitude limits.

Bands mosaicing & stacking. After that, for every band we combine all framelets into a band mosaic. Finally, we combine the mosaics of individual bands into one multi-band mosaic and mask locations that do not have all bands.

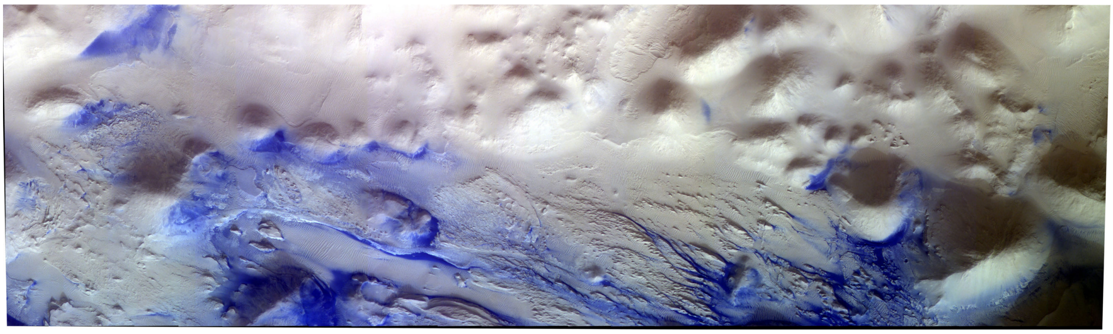
6.6.2 Results

In this section, we present a qualitative and quantitative analysis of CaSSIS color mosaics reconstructed using the method described in the previous section. For the qualitative analysis, in Figure 6.14 we show color mosaics for sequences from “MTP07”, “STP027” and “boot2”, summarized in Table 6.5. We reconstruct the mosaics using the refined CaSSIS parameters and the BA. In Figure 6.15 we show close-ups of these mosaics and, for comparison, same mosaics reconstructed with the nominal CaSSIS parameters and without the BA. Note, that, the mosaics reconstructed with the refined parameters and BA do not have color fringes.

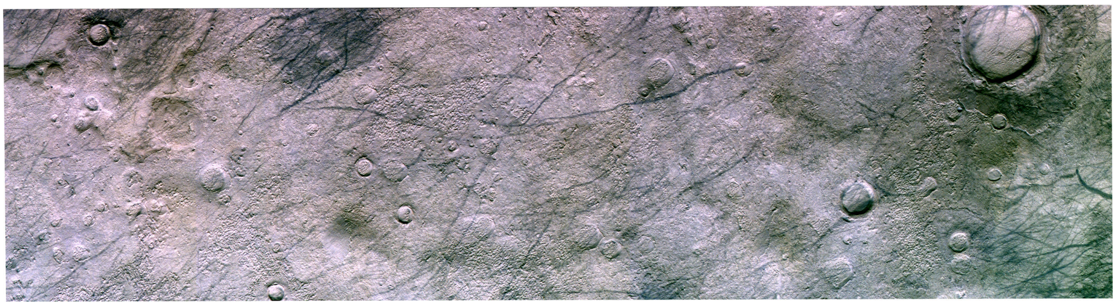
As a quantitative measure of color mosaic quality, we use spatial misalignment between the “NIR” and the “PAN” bands. We use these band because they are present in almost all sequences. To compute the misalignment for a particular color mosaic, we perform sub-pixel block matching between “NIR” and “PAN” bands for sparse regularly sampled locations and compute median of the along-track, across-track and L^2 misalignment. In Table 6.6 we show average misalignment for 26 reconstructed color mosaics from “MTP07”, “STP027” and “boot2” (we use all sequences that have “NIR” and “PAN” bands).

The color mosaics reconstructed with the refined camera parameters have significantly smaller bands misalignment than the color mosaics reconstructed with the nominal parameters. In

MY34_004060_181_1



MY34_004060_228_1



MY34_004071_200_1

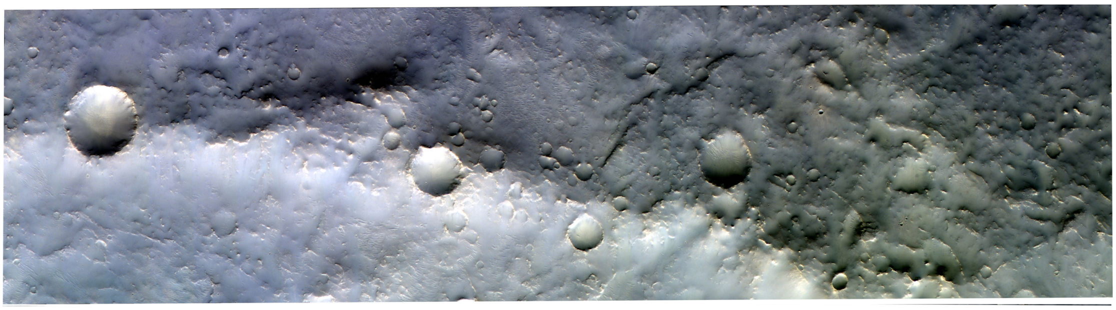


Figure 6.14 – Examples of color mosaics reconstructed using the proposed method, the refined CaSSIS parameters and BA. We show the mosaics in false colour (NIR → red, PAN → green, BLU → blue channel).

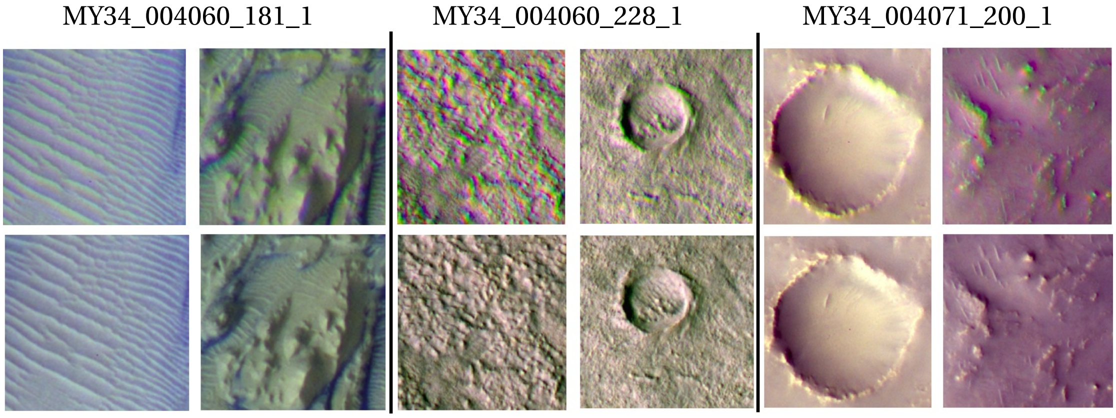


Figure 6.15 – Close-ups of color mosaics from Figure 6.14. The mosaics in the first row we reconstruct using the nominal CaSSIS parameters and the mosaics in the second row – using the refined parameters and BA. Note, that the mosaics in the second row do not have color fringes. We show the mosaics in false colour (NIR → red, PAN → green, BLU → blue channel).

the former case, the remaining misalignment is due to a non-systematic CaSSIS orientation error, caused by the camera jitter. The presence of the jitter becomes obvious from Figure 6.16, where we show pixel-wise bands misalignment for mosaics reconstructed with the refined camera parameters but without the BA. Note, that the misalignment varies from framelet to framelet which indicates the presence of the camera jitter. As we show in Table 6.6, these non-systematic pointing errors can be reduced by the proposed BA.

6.7 Stereo reconstruction

In this section, in § 6.7.1 we propose a tool for reconstruction of DTMs from raw CaSSIS data and in § 6.7.2 we provide a qualitative and quantitative analysis of their quality. We develop this tool in Python using USGS ISIS [200] and Ames Stereo Pipeline (ASP) [202] and make the source code and standalone package for Linux available online[3]. The standalone package we build using the CDE virtualization tool [207] and it contains all dependencies.

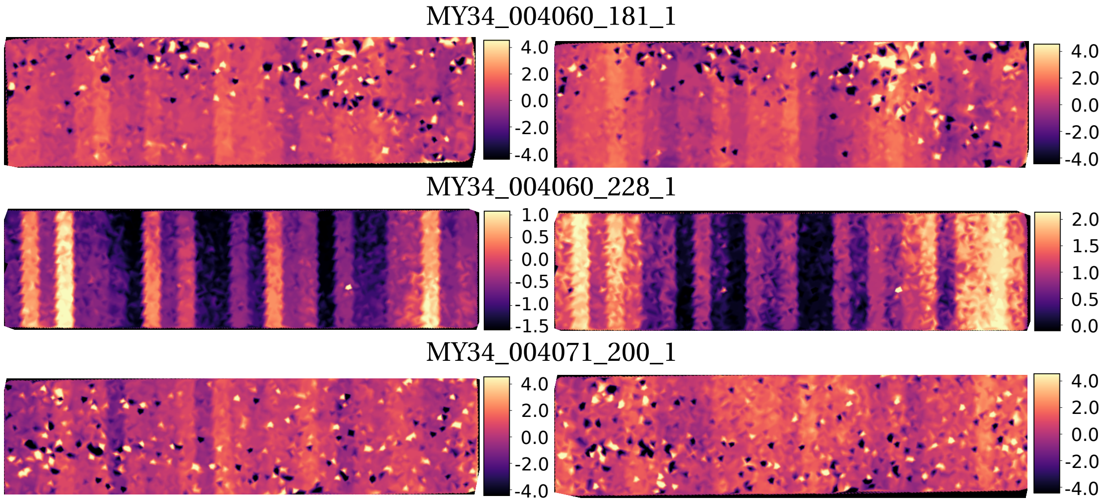
6.7.1 Method

The method takes as an input stereo observation that consists of two CaSSIS sequences and produces raster with heights above the Mars ellipsoid, called *Digital Terrain Model (DTM)*. The DTM reconstruction method consists of three steps described below.

Stereo sequences mosaicing. First, we reconstruct color mosaics for both sequences from a stereo observation. For this, we rely on the procedure described in § 6.6 with two minor modifications. The first modification is that we use the same projection parameters for both

---

[3] https://github.com/tlkvstepan/CaSSIS_color_and_stereo



(a) Across-track misalignment

(b) Along-track misalignment

Figure 6.16 – Pixel-wise bands misalignment in the color mosaics reconstructed with the refined camera parameters but without the BA. We compute the misalignment between “PAN” and “NIR” bands. The dark and the bright speckles correspond to registration errors. Notice, that the misalignment varies from framelet to framelet, which indicates the presence of the camera jitter.

sequences to ensure that the resulting mosaics have the same resolution and lay on the same coordinate grid. Another modification is that we include tracing layers to the mosaics, that allows finding the origin of every pixel in the mosaics. We use this information later, during stereo triangulation.

Stereo matching. Next, we perform a stereo matching between the sequences. Since for the map-projected mosaics the epipolar geometry described in § 2.1.3 is not valid, we perform the stereo matching in 2d space. For that, we rely on Ames Stereo Pipeline (ASP) [202], which performs: 2d coarse-to-fine *correlation* followed by *sub-pixel refinement* and *outliers rejection*. Finally, it produces files with horizontal and vertical displacements between the mosaics.

Stereo triangulation. Finally, we perform a stereo triangulation to find positions of the mosaic points in 3d space. Because ASP is not capable of performing the triangulation for map-projected mosaics, we implement it ourselves in C++ using USGS ISIS API [200]. Since USGS ISIS does not preserve the camera information required for the triangulation in the mosaics, we trace the origin of every pixel back to the corresponding map-projected framelet, where this information is present. For every resulting 3d point, we find the closest point on Mars ellipsoid, corresponding geographic coordinates, and elevation with respect to the ellipsoid. To compute the elevations for the original coordinate grid we use simple nearest neighbourhood interpolation.

Table 6.7 – Summary of CaSSIS observations that we use for qualitative and quantitative analysis of DTMs. Note, that we show the overlapping HiRISE products in Table 6.8 in the same order.

Observation	**Acquisition date and time**	**Center (lat, lon), [deg]**	**Resolution, [m/px]**
MY34_004219_201	03-11-2018 at 09:33:06	(-18.617, 62.619)	4.4
MY34_004069_162	16-11-2018 at 21:39:57	(18.601, 77.416)	4.7
MY34_004041_196	19-10-2018 at 19:54:54	(-14.615, 175.641)	4.5

Table 6.8 – Summary of HiRISE DTMs that we use as ground truth for quantitative analysis of CaSSIS DTMs. Note, that we show the overlapping CaSSIS observations in Table 6.7 in the same order.

DTM product	**Resolution, [m/px]**	**MOLA RMS, [m]**
DTEED_021494_1610_013134_1610_A01	2.0	4.610
DTEEC_002387_1985_003798_1985_A01	1.0	4.773
DTEEC_036087_1655_035164_1655_U01	1.0	-

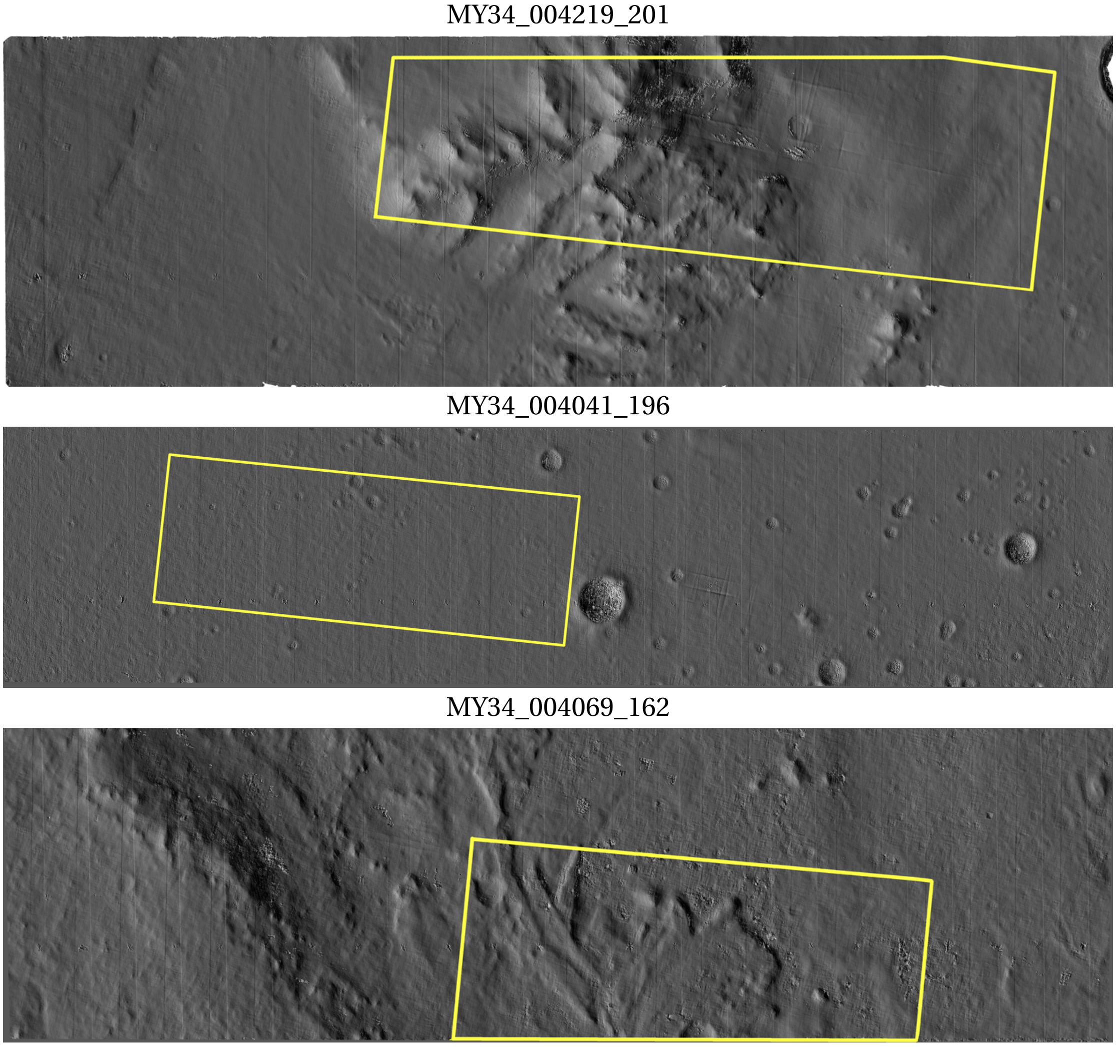
6.7.2 Results

In this section, we present a qualitative and quantitative analysis of CaSSIS DTMs reconstructed using the proposed method.

For experiments, we select three CaSSIS stereo observations overlapping with officially released DTMs for High-Resolution Imaging Science Experiment (HiRISE) on-board of Mars Reconnaissance Orbiter (MRO) [208] available from Planetary Data System (PDS) archive [209]. We summarize these overlapping observations in Tables 6.7 and 6.8. We use HiRISE DTMs as a ground truth, because they have a much higher spatial resolution (≈ 1 m/px) than CaSSIS DTMs (≈ 4.5 m/px) and because they are aligned with Mars Orbiter Laser Altimeter (MOLA) data and manually validated by a human expert.

For the qualitative analysis, we visualize reconstructed CaSSIS DTMs in Figure 6.17. The DTMs are reconstructed automatically using PAN channels of color mosaics. For each DTM, we show a synthesized shade image, which is called *hillshade*. The reconstructed CaSSIS DTMs have several apparent artifacts such as elevation bumps on framelets boundaries, caused by remaining orientation errors, spurious line patterns, probably caused by reflections from camera elements, and depth outliers, caused by stereo matching errors.

For the quantitative analysis, we compute the average absolute vertical error of the reconstructed DTMs relative to the corresponding HiRISE DTMs. For that, we match projection parameters and datum of the HiRISE DTMs to these of the CaSSIS DTMs and align HiRISE and CaSSIS DTMs using *Iterative Closest Point (ICP)* algorithm implemented in the ASP [202]. We initialize the algorithm with 5-10 manually matched points. We perform the alignment using only 3d translation. After the alignment, we compare the DTMs and show the results


MY34_004219_201

MY34_004041_196

MY34_004069_162

Figure 6.17 – Hillshade images synthesized from the CaSSIS DTMs reconstructed using the proposed method. The DTMs are reconstructed automatically using the “PAN” band. The yellow contours show areas of overlap with the corresponding HiRISE DTMs. The reconstructed DTMs have several apparent artifacts such as elevation bumps on framelets boundaries, caused by remaining orientation errors, spurious line patterns, probably caused by reflections from camera elements, and depth outliers, caused by stereo matching errors.

Table 6.9 – Average absolute vertical errors with standard deviations for CaSSIS DTMs. We compute the errors relative to the corresponding HiRISE DTMs after the alignment using 3d translations. The aligning translations are large and very similar for different observations, which suggests the presence of a significant camera pointing error, perhaps due to the incorrect image timing. After the alignment, the errors are relatively small.

Observation	**Absolute vertical error, [m]**	**Aligning translation, [m]** East (across-track)	North (along-track)	Down
MY34_004219_201	6.0 ± 4.0	-813.8	3489.1	218.3
MY34_004069_162	5.4 ± 14.1	-822.9	3243.8	-231.0
MY34_004041_196	2.2 ± 1.7	-762.2	3309.2	-191.8

in Table 6.9. Note, that the translation vectors, aligning the point clouds are large, especially along-track. This probably indicates the presence of a significant camera pointing error, perhaps due to the incorrect image timing. After the alignment, the absolute vertical errors are relatively small. This is in agreement with observations in [210], where the authors perform the same comparison using different tools.

6.8 Conclusion

In this work, we propose a novel method for geometric calibration of a telescope with a large focal length and complex optical distortion. We apply this method to refine the nominal camera parameters of the CaSSIS and show that the refined parameters significantly improve the quality of CaSSIS color mosaics. The refined parameters became part of the official ExoMars Trace Gas Orbiter SPICE kernels [201] and USGS ISIS [200] releases. We also develop automatic tools for reconstruction of color mosaics and DTMs from raw CaSSIS data and experimentally show that color bands misalignment in the reconstructed color mosaics and average absolute vertical error of the reconstructed DTMs are relatively small. However, qualitative analysis of CaSSIS DTMs revealed multiple artifacts in DTMs, probably caused by an incorrect stereo matching, imperfect camera pointing and possibly, internal reflection. These problems should be addressed in future works. The practical significance of this work is supported by the fact that its results are widely used by a scientific community [181, 182, 210–214].

7 Conclusions

In this chapter, in § 7.1 we summarize practical and scientific contributions of this dissertation and in § 7.2 discuss possible future research direction.

7.1 Summary

In the “Research” part (Part I) of this work, we focus on deep learning-based methods for stereo reconstruction which discover multiple depth cues directly from training data and on their potential application in satellite stereo imagery. The main scientific contributions of the “Research” part are following.

In Chapter 3, based on [69], we show that it is possible to learn a high quality deep metric for stereo matching without any labeled training data, relying only on coarse information about scenes geometry and a stereo system. The proposed weakly supervised method alternatively computes matches using the current metric, regularizes them and uses as ground truth for metric learning. For a given network architecture, training with this method without ground truth produces a metric with the performance as good as the performance of the same metric trained with the said ground truth. The method allows training a deep metric when labeled training data is not available or contaminated with noise.

Next, in Chapter 4, based on [173], we show that using domain knowledge it is possible to design a more practical end-to-end deep stereo method. We propose a novel bottleneck module that drastically reduces the memory footprint of the network, allowing it to leverage a larger spatial context to resolve matching ambiguities and process larger images. Also, we proposed a new loss and estimator that makes the method less sensitive to ambiguous matches, and applicable to any disparity range without retraining. At the time of the article submission, the method was among the top performers across the benchmarks.

In Chapter 5, based on [215], we demonstrate that our method from § 4 developed for a frame-based stereo system can be easily re-purposed for an event-based stereo system, producing an asynchronous stream of events, triggered by significant pixel intensity changes. For that, we introduce a novel module for event sequence embedding which builds a representation of a sequence by firstly aggregating information locally across time, using a novel fully-connected layer for an irregularly sampled continuous domain, and then across discrete spatial domain. The resulting method is the first learning-based stereo method for an event-based camera and the only method that produces dense results, moreover, it outperforms all previous methods on Multi-Vehicle Stereo Event Camera Dataset (MVSEC) benchmark by a large margin.

In the "Applied" part (Part II) of this work, we focus on geometric calibration and developing automatic stereo reconstruction and mosaicing tools for the CaSSIS. The practical contributions of the "Applied" part are following.

In Chapter 5 we develop a new geometric calibration method for a satellite telescope with a large focal length and complex optical distortion. To our knowledge, this is the first calibration method that uses hundreds of images of starfields to robustly estimate intrinsic and extrinsic parameters of the camera and a complex optical distortion model. With the proposed method we refine nominal parameters of the CaSSIS and enabled reconstruction of high quality color mosaics without color fringes and Digital Terrain Models. These refined parameters became part of the official Trace Gas Orbiter SPICE kernels [201] and USGS ISIS [200] releases. We also develop and validate tools for an automatic reconstruction of color mosaics and DTMs from a raw CaSSIS data, which are widely used by a scientific community [181, 182, 210–214].

7.2 Future directions

The main weakness of the weakly supervised method proposed in Chapter 3 is its reliance on problem constraints identified and hard-coded by a human into unsupervised losses. These losses probably do not account for all constraints and most of them require careful parameter tuning. It would be interesting to explore if they can be learned directly from a data in a form of *error detector network* and then used for a fine-tuning of a stereo network using unlabeled target domain examples. Since error detection is typically easy, we expect that the error detector will require relatively small capacity. It can be trained on a source domain in a domain adversarial manner [118], or on a few labeled examples from the target domain.

For the stereo method from the Chapter 4 one major problem is a large number of parameters and memory footprint of 3d convolutions in the regularization network. Moreover, to process map-projected satellite stereo images we will have to use 4d convolutions instead of 3d for a 2d search in horizontal and vertical directions, which will make the problem even worse. It would be interesting to explore different parametrizations of these convolutions, that have a smaller number of parameters and permit memory-efficient processing. One can, for example, use separable convolutions [216, 217].

There is a room for improvement for an event sequence embedding method from the Chapter 5. We showed that in the presence of a large spatial context, the network starts to ignore the temporal context. We suspect that this happens because the dataset which we use for training, composed of videos acquire from a quadcopter, contains a lot of abrupt motions that render the temporal context less reliable than the spatial. However, we believe that for frames with less motion, the temporal context can still be beneficial. It would be interesting to explore a mechanism, which will be able to decide about the appropriate amount of spatial and temporal context on a per-location basis. We believe that it can be based on the self-attention mechanism, described in [218, 219]. Also it will be interesting to apply the proposed temporal aggregation method to other problems, that require large temporal context, such as optic flow and single view depth estimation.

Finally, the calibration method and the stereo reconstruction tool proposed in § 6 also can be improved. Regarding the calibration, during the experiments, we noticed that there is a significant translation between CaSSIS mosaics and mosaics from other instruments. This suggests the presence of a large camera pointing error of unknown origin. We suspect that it might be related to an error in image acquisition timestamp. This might explain why the translation is mostly along the track and why the pointing error is not presented in starfield images. This problem should be investigated in future works. Regarding the stereo reconstruction, we found several artifacts in reconstructed DTMs which should be fixed in future works. The most prominent stair-step problem on framelets stitches, caused by a small pointing error, can be alleviated by performing bundle adjustment jointly for a whole stereo observation, using correspondences between stereo observation established by interest point matching [49] and using all color bands in stereo triangulation.

Finally, the most rewarding future work is to apply the results of our research to CaSSIS data. This would require redesigning the network from Chapter 4 to preform 2d stereo matching because map-projected mosaics do not have epipolar geometry. The resulting network can be pre-trained on a synthetic optic flow dataset, such as [74] and then fine-tuned on real CaSSIS images without ground truth using our weakly supervised method from Chapter 3, also redesigned for a 2d stereo matching.

Bibliography

[1] J. Canny, “A Computational Approach to Edge Detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, no. 6, pp. 679–698, 1986.

[2] C. Tomasi and R. Manduchi, “Bilateral Filtering for Gray and Color Images,” in *International Conference on Computer Vision (ICCV)*, vol. 98, p. 2, 1998.

[3] D. J. Jobson, Z.-u. Rahman, and G. A. Woodell, “Properties and performance of a center/surround retinex,” *IEEE Transactions on Image Processing*, vol. 6, no. 3, pp. 451–462, 1997.

[4] B. D. Lucas, T. Kanade, *et al.*, “An iterative image registration technique with an application to stereo vision,” *Proceedings of International Joint Conference on Artificial Intelligence*, pp. 674–679, 1981.

[5] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3061–3070, 2015.

[6] K. W. Nam, J. Park, I. Y. Kim, and K. G. Kim, “Application of stereo-imaging technology to medical field,” *Healthcare Informatics Research*, vol. 18, no. 3, pp. 158–163, 2012.

[7] D. E. Shean, O. Alexandrov, Z. M. Moratto, B. E. Smith, I. R. Joughin, C. Porter, and P. Morin, “An automated, open-source pipeline for mass production of digital elevation models (DEMs) from very-high-resolution commercial stereo satellite imagery,” *Journal of Photogrammetry and Remote Sensing*, vol. 116, pp. 101–117, 2016.

[8] T.-C. Wang, M. Srikanth, and R. Ramamoorthi, “Depth from Semi-Calibrated Stereo and Defocus,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3717–3726, 2016.

[9] J. T. Barron, A. Adams, Y. Shih, and C. Hernández, “Fast bilateral-space stereo for synthetic defocus,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 4466–4474, 2015.

[10] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár, “Learning to Refine Object Segments,” in *European Conference on Computer Vision (ECCV)*, pp. 75–91, Springer, 2016.

[11] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, “Convolutional pose machines,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 4724–4732, 2016.

[12] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.

[13] A. Brock, J. Donahue, and K. Simonyan, “Large Scale GAN Training for High Fidelity Natural Image Synthesis,” *Computing Research Repository (CoRR)*, 2018.

[14] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, “Show and Tell: A Neural Image Caption Generator,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3156–3164, 2015.

[15] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, “Ask Me Anything: Dynamic Memory Networks for Natural Language Processing,” in *Proceedings of International Conference on Machine Learning (ICML)*, pp. 1378–1387, JMLR. org, 2016.

[16] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah, “Shape-from-shading: a survey,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 21, no. 8, pp. 690–706, 1999.

[17] P. Favaro and S. Soatto, “A geometric approach to shape from defocus,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 3, pp. 406–417, 2005.

[18] A. P. Witkin, “Recovering surface shape and orientation from texture,” *Artificial intelligence*, vol. 17, no. 1-3, pp. 17–45, 1981.

[19] J. Žbontar and Y. LeCun, “Computing the Stereo Matching Cost With a Convolutional Neural Network,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 1592–1599, 2015.

[20] S. Sukhbaatar and R. Fergus, “Learning from Noisy Labels with Deep Neural Networks,” *International Conference on Learning Representations Workshop (ICLRW)*, 2014.

[21] P. Fischer, A. Dosovitskiy, and T. Brox, “Descriptor Matching with Convolutional Neural Networks: a Comparison to SIFT,” *Computing Research Repository (CoRR)*, 2014.

[22] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, 2003.

[23] D. Brown, “Decentering Distortion of Lenses,” *Photogrammetric Engineering*, vol. 32, pp. 444–462, May 1966.

[24] T. Kanade and M. Okutomi, “A stereo matching algorithm with an adaptive window: Theory and experiment,” in *IEEE International Conference on Robotics and Automation*, pp. 1088–1095, IEEE, 1991.

[25] O. Veksler, “Fast variable window for stereo correspondence using integral images,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–I, IEEE, 2003.

[26] A. Hosni, M. Bleyer, M. Gelautz, and C. Rhemann, “Local stereo matching using geodesic support weights,” in *IEEE International Conference on Image Processing (ICIP)*, pp. 2093–2096, IEEE, 2009.

[27] K. Yamaguchi, T. Hazan, D. McAllester, and R. Urtasun, “Continuous Markov Random Fields for Robust Stereo Estimation,” in *European Conference on Computer Vision (ECCV)*, pp. 45–58, Springer, 2012.

[28] J. Sun, N.-N. Zheng, and H.-Y. Shum, “Stereo Matching Using Belief Propagation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, no. 7, pp. 787–800, 2003.

[29] D. Scharstein and C. Pal, “Learning Conditional Random Fields for Stereo,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, IEEE, 2007.

[30] P. Knöbelreiter, C. Reinbacher, A. Shekhovtsov, and T. Pock, “End-to-End Training of Hybrid CNN-CRF Models for Stereo,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 2339–2348, 2017.

[31] P. N. Belhumeur, “A Bayesian approach to binocular steropsis,” *International Journal of Computer Vision (IJCV)*, vol. 19, no. 3, pp. 237–260, 1996.

[32] Y. Ohta and T. Kanade, “Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, no. 2, pp. 139–154, 1985.

[33] D. Marr and T. Poggio, “Cooperative computation of stereo disparity,” *From the Retina to the Neocortex*, pp. 239–243, 1976.

[34] H. Hirschmuller, “Accurate and efficient stereo processing by semi-global matching and mutual information,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 807–814, IEEE, 2005.

[35] A. Hosni, C. Rhemann, M. Bleyer, C. Rother, and M. Gelautz, “Fast Cost-Volume Filtering for Visual Correspondence and Beyond,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 35, no. 2, pp. 504–511, 2012.

[36] O. Veksler, “Stereo correspondence by dynamic programming on a tree,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 2, pp. 384–390, IEEE, 2005.

[37] V. Kolmogorov and R. Zabih, “Computing Visual Correspondence with Occlusions via Graph Cuts,” *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, vol. 2, pp. 508–515, 2001.

[38] O. Woodford, P. Torr, I. Reid, and A. Fitzgibbon, “Global stereo reconstruction under second-order smoothness priors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 12, pp. 2115–2128, 2009.

[39] A. Klaus, M. Sormann, and K. Karner, “Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure,” in *International Conference on Pattern Recognition*, vol. 3, pp. 15–18, IEEE, 2006.

[40] F. Güney and A. Geiger, “Displets: Resolving Stereo Ambiguities using Object Knowledge,” *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4165–4175, 2015.

[41] K. Yamaguchi, D. Mcallester, and R. Urtasun, “Efficient Joint Segmentation , Occlusion Labeling, Stereo and Flow Estimation,” *European Conference on Computer Vision (ECCV)*, pp. 756–771, 2014.

[42] A. O. Ulusoy, M. J. Black, and A. Geiger, “Semantic multi-view stereo: Jointly estimating objects and voxels,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4531–4540, IEEE, 2017.

[43] H. Hirschmuller and D. Scharstein, “Evaluation of Cost Functions for Stereo Matching,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2007.

[44] H. Hirschm, “Evaluation of Stereo Matching Costs on Images with Radiometric Differences,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 31, no. 9, pp. 1582–1599, 2008.

[45] H. Hirschmüller, P. R. Innocent, and J. Garibaldi, “Real-Time Correlation-Based Stereo Vision with Reduced Border Errors,” *International Journal of Computer Vision (IJCV)*, vol. 47, no. 1-3, pp. 229–246, 2002.

[46] A. Ansar, A. Castano, and L. Matthies, “Enhanced Real-time Stereo Using Bilateral Filtering,” *Proceedings of International Symposium on 3D Data Processing*, pp. 455–462, 2004.

[47] R. Zabih and J. Woodfill, “Non-parametric Local Transforms for Computing Visual Correspondence,” *European Conference on Computer Vision (ECCV)*, pp. 151–158, 1994.

[48] J. Kim, V. Kolmogorov, and R. Zabih, “Visual Correspondence Using Energy Minimization and Mutual Information,” in *Proceedings of IEEE International Conference on Computer Vision (ICCV)*, pp. 1033–1040, IEEE, 2003.

[49] D. G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision (IJCV)*, vol. 60, no. 2, pp. 91–110, 2004.

[50] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, “Speeded-Up Robust Features (SURF),” *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

[51] M. Calonder, V. Lepetit, M. Özuysal, T. Trzcinski, C. Strecha, and P. Fua, “BRIEF: Computing a local binary descriptor very fast,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 7, pp. 1281–1298, 2011.

[52] E. Tola, V. Lepetit, and P. Fua, “A fast local descriptor for dense matching,” *IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1–8, 2008.

[53] K. Mikolajczyk, K. Mikolajczyk, C. Schmid, and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 27, no. 10, p. 1615, 2005.

[54] T. Trzcinski, M. Christoudias, V. Lepetit, and P. Fua, “Learning Image Descriptors with the Boosting-Trick,” *Advances in Neural Information Processing Systems (NIPS)*, pp. 269–277, 2012.

[55] K. Simonyan, A. Vedaldi, and A. Zisserman, “Descriptor learning using convex optimisation,” *European Conference on Computer Vision (ECCV)*, pp. 243–256, 2012.

[56] C. Strecha, A. M. Bronstein, M. M. Bronstein, and P. Fua, “LDAHash: Improved matching with smaller descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 1, pp. 66–78, 2011.

[57] W. Kim, J. Park, and K. M. Lee, “Stereo matching using population-based MCMC,” *International Journal of Computer Vision (IJCV)*, vol. 83, no. 2, pp. 195–209, 2009.

[58] H. Hirschmuller, “Stereo Processing by Semi-Global Matching and Mutual Information,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 30, no. 2, pp. 328–341, 2007.

[59] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “FlowNet: Learning Optical Flow with Convolutional Networks,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015.

[60] X. Mei, X. Sun, M. Zhou, S. Jiao, H. Wang, and X. Zhang, “On building an accurate stereo matching system on graphics hardware,” in *IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 467–474, IEEE, 2011.

[61] M. Jahrer, M. Grabner, and H. Bischof, “Learned local descriptors for recognition and matching,” *Computer Vision Winter Workshop*, vol. 2, no. 3, 2008.

[62] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer, “Discriminative Learning of Deep Convolutional Feature Point Descriptors,” *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 118–126, 2015.

[63] S. Zagoruko and N. Komodakis, “Learning to Compare Image Patches via Convolutional Neural Networks,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 4353–4361, 2015.

[64] Xufeng Han, T. Leung, Y. Jia, R. Sukthankar, and A. C. Berg, “MatchNet: Unifying feature and metric learning for patch-based matching,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3279–3286, 2015.

[65] B. G. V. Kumar, G. Carneiro, and I. Reid, “Learning Local Image Descriptors with Deep Siamese and Triplet Convolutional Networks by Minimising Global Loss Functions,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 5385–5394, 2016.

[66] J. Zbontar and Y. LeCun, “Stereo matching by training a convolutional neural network to compare image patches,” *Journal of Machine Learning Research (JMLR)*, vol. 17, no. 1-32, p. 2, 2016.

[67] Z. Chen, X. Sun, and L. Wang, “A Deep Visual Correspondence Embedding Model for Stereo Matching Costs,” *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 972–980, 2015.

[68] J. Bromley, I. Guyon, Y. Lecun, E. Säckinger, and R. Shah, “Signature Verification using a "Siamese" Time Delay Neural Network,” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 737–744, 1994.

[69] S. Tulyakov, A. Ivanov, and F. Fleuret, “Weakly supervised learning of deep metrics for stereo reconstruction,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1348–1357, 2017.

[70] V. Balntas, E. Johns, L. Tang, and K. Mikolajczyk, “PN-Net: Conjoined Triple Deep Network for Learning Local Image Descriptors,” *Computing Research Repository (CoRR)*, 2016.

[71] A. Seki and M. Pollefeys, “SGM-Nets: Semi-global matching with neural networks,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 231–240, 2017.

[72] A. Shaked and L. Wolf, “Improved Stereo Matching with Constant Highway Networks and Reflective Confidence Learning,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 4641–4650, 2017.

[73] S. Gidaris and N. Komodakis, “Detect, Replace, Refine: Deep Structured Prediction For Pixel Wise Labeling,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 5248–5257, 2017.

[74] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, “A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow

Estimation," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 4040–4048, 2016.

[75] A. Kendall, H. Martirosyan, S. Dasgupta, P. Henry, R. Kennedy, A. Bachrach, and A. Bry, "End-to-End Learning of Geometry and Context for Deep Stereo Regression," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 66–75, 2017.

[76] Y. Zhong, Y. Dai, and H. Li, "Self-Supervised Learning for Stereo Matching with Self-Improving Ability," *Computing Research Repository (CoRR)*, 2017.

[77] J. Pang, W. Sun, J. Ren, C. Yang, and Q. Yan, "Cascade Residual Learning: A Two-stage Convolutional Neural Network for Stereo Matching," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 887–895, 2017.

[78] Z. Jie, P. Wang, Y. Ling, B. Zhao, Y. Wei, J. Feng, and W. Liu, "Left-Right Comparative Recurrent Model for Stereo Matching," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3838–3846, 2018.

[79] Z. Liang, Y. Feng, Y. G. H. L. W. Chen, and L. Q. L. Z. J. Zhang, "Learning for Disparity Estimation through Feature Constancy," *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 2811–2820, 2018.

[80] J.-R. Chang and Y.-S. Chen, "Pyramid Stereo Matching Network," *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 5410–5418, 2018.

[81] A. Pilzer, D. Xu, M. Puscas, E. Ricci, and N. Sebe, "Unsupervised adversarial depth estimation using cycled generative networks," in *International Conference on 3D Vision*, pp. 587–595, IEEE, 2018.

[82] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia, "SegStereo: Exploiting Semantic Information for Disparity Estimation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 636–651, 2018.

[83] X. Song, X. Zhao, L. Fang, and H. Hu, "EdgeStereo: An Effective Multi-Task Learning Network for Stereo Matching and Edge Detection," *Computing Research Repository (CoRR)*, 2019.

[84] B. Babenko, "Multiple Instance Learning: Algorithms and Applications," *PubMed-NCBI*, pp. 1–19, 2008.

[85] J. Wu, Y. Yu, C. Huang, and K. Yu, "Deep Multiple Instance Learning for Image Classification and Auto-Annotation," *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3460–3469, 2015.

[86] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2223–2232, 2017.

[87] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, “Extracting and composing robust features with denoising autoencoders,” in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pp. 1096–1103, ACM, 2008.

[88] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, “Context Encoders: Feature Learning by Inpainting,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, 2016.

[89] R. Zhang, P. Isola, and A. A. Efros, “Colorful Image Colorization,” in *European Conference on Computer Vision (ECCV)*, pp. 649–666, Springer, 2016.

[90] M. Noroozi, H. Pirsiavash, and P. Favaro, “Representation learning by learning to count,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 5898–5906, 2017.

[91] C. Doersch, A. Gupta, and A. A. Efros, “Unsupervised visual representation learning by context prediction,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1422–1430, 2015.

[92] S. Gidaris, P. Singh, and N. Komodakis, “Unsupervised representation learning by predicting image rotations,” *Computing Research Repository (CoRR)*, 2018.

[93] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[94] I. Triguero, S. García, and F. Herrera, “Self-labeled techniques for semi-supervised learning: taxonomy, software and empirical study,” *Knowledge and Information Systems*, vol. 42, no. 2, pp. 245–284, 2015.

[95] S. E. Reed and H. Lee, “Training deep neural networks on noisy labels with bootstrapping,” *Computing Research Repository (CoRR)*, 2015.

[96] K. All, D. Hasler, and F. Fleuret, “FlowBoost - Appearance learning from sparsely annotated video,” in *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1433–1440, IEEE Computer Society, 2011.

[97] C. Tan, F. Sun, T. Kong, W. Zhang, C. Yang, and C. Liu, “A survey on deep transfer learning,” in *International Conference on Artificial Neural Networks*, pp. 270–279, Springer, 2018.

[98] M. Wang and W. Deng, “Deep Visual Domain Adaptation: A Survey,” *Neurocomputing*, vol. 312, pp. 135–153, 2018.

[99] S. J. Pan and Q. Yang, “A Survey on Transfer Learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, 2009.

[100] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, “The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3234–3243, 2016.

[101] C. Zhou, H. Zhang, X. Shen, and J. Jia, "Unsupervised learning of stereo matching," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1567–1575, 2017.

[102] A. Tonioni, M. Poggi, S. Mattoccia, and L. Di Stefano, "Unsupervised Adaptation for Deep Stereo," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1605–1613, 2017.

[103] M. Poggi and S. Mattoccia, "Learning from scratch a confidence measure," in *British Machine Vision Conference (BMVC)*, 2016.

[104] R. Garg, V. K. BG, G. Carneiro, and I. Reid, "Unsupervised cnn for single view depth estimation: Geometry to the rescue," in *European Conference on Computer Vision (ECCV)*, pp. 740–756, Springer, 2016.

[105] J. Nath Kundu, P. Krishna Uppala, A. Pahuja, and R. Venkatesh Babu, "AdaDepth: Unsupervised Content Congruent Adaptation for Depth Estimation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2656–2665, 2018.

[106] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 270–279, 2017.

[107] S. Pillai, R. Ambruş, and A. Gaidon, "SuperDepth: Self-Supervised, Super-Resolved Monocular Depth Estimation," in *International Conference on Robotics and Automation (ICRA)*, pp. 9250–9256, IEEE, 2019.

[108] A. Atapour-Abarghouei and T. P. Breckon, "Real-time monocular depth estimation using synthetic data with domain adaptation via image style transfer," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2800–2810, 2018.

[109] A. Tonioni, F. Tosi, M. Poggi, S. Mattoccia, and L. D. Stefano, "Real-time self-adaptive deep stereo," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 195–204, 2019.

[110] J. Pang, W. Sun, C. Yang, J. Ren, R. Xiao, J. Zeng, and L. Lin, "Zoom and learn: Generalizing deep stereo matching to novel domains," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2070–2079, 2018.

[111] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1851–1858, 2017.

[112] A. Gordon, H. Li, R. Jonschkowski, and A. Angelova, "Depth from Videos in the Wild: Unsupervised Monocular Depth Learning from Unknown Cameras," *Computing Research Repository (CoRR)*, 2019.

[113] H. Zhan, R. Garg, C. Saroj Weerasekera, K. Li, H. Agarwal, and I. Reid, “Unsupervised learning of monocular depth estimation and visual odometry with deep feature reconstruction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 340–349, 2018.

[114] Z. Yin and J. Shi, “GeoNet: Unsupervised Learning of Dense Depth, Optical Flow and Camera Pose,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1983–1992, 2018.

[115] M. Jaderberg, K. Simonyan, A. Zisserman, *et al.*, “Spatial Transformer Networks,” in *Advances in Neural Information Processing Systems (NIPS)*, pp. 2017–2025, 2015.

[116] Y. Kuznietsov, J. Stuckler, and B. Leibe, “Semi-supervised deep learning for monocular depth map prediction,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6647–6655, 2017.

[117] Y. Wang, Y. Yang, Z. Yang, L. Zhao, P. Wang, and W. Xu, “Occlusion aware unsupervised learning of optical flow,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4884–4893, 2018.

[118] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, “Domain-adversarial training of neural networks,” *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[119] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 3354–3361, IEEE, 2012.

[120] D. Scharstein and R. Szeliski, “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms,” *International Journal of Computer Vision (IJCV)*, vol. 47, no. 1-3, pp. 7–42, 2002.

[121] D. Scharstein and R. Szeliski, “High-accuracy stereo depth maps using structured light,” *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–I, 2003.

[122] D. Scharstein, H. Hirschmüller, Y. Kitajima, G. Krathwohl, N. Nešić, X. Wang, and P. Westling, “High-resolution stereo datasets with subpixel-accurate ground truth,” in *German Conference on Pattern Recognition*, pp. 31–42, Springer, 2014.

[123] “KITTI 2012 and 2015 scoreboards.” http://www.cvlibs.net/datasets/kitti/, accessed on November 14th, 2016.

[124] “Middlebury scoreboard.” http://vision.middlebury.edu/stereo/eval3/, accessed on November 14th, 2016.

[125] D. Marr and T. Poggio, "A Computational Theory of Human Stereo Vision," *Proceedings of the Royal Society of London, Series B. Biological Sciences*, vol. 204, no. 1156, pp. 301–328, 1979.

[126] R. Collobert, K. Kavukcuoglu, and C. Farabet, "Torch7: A Matlab-like Environment for Machine Learning," in *BigLearn, Advances in Neural Information Processing Systems Workshop*, 2011.

[127] "Torch7 deep learning framework." http://torch.ch/, accessed on June 2nd, 2010.

[128] K.-R. Kim and C.-S. Kim, "Adaptive smoothness constraints for efficient stereo matching using texture and edge information," in *IEEE International Conference on Image Processing (ICIP)*, pp. 3429–3433, IEEE, 2016.

[129] J. T. Barron and B. Poole, "The Fast Bilateral Solver," in *European Conference on Computer Vision (ECCV)*, pp. 617–632, Springer, 2016.

[130] A. Li, D. Chen, Y. Liu, and Z. Yuan, "Coordinating Multiple Disparity Proposals for Stereo Computation," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 4022–4030, 2016.

[131] C. Zhang and Z. Li, "MeshStereo : A Global Stereo Model with Mesh Alignment Regularization for View Interpolation," *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2057–2065, 2015.

[132] A. Seki and M. Pollefeys, "Patch Based Confidence Prediction for Dense Disparity Map," in *British Machine Vision Conference (BMVC)*, vol. 2, p. 4, 2016.

[133] V. Ntouskos and F. Pirri, "Confidence driven TGV fusion," *Computing Research Repository (CoRR)*, 2016.

[134] C. Vogel, K. Schindler, and S. Roth, "3D Scene Flow Estimation with a Piecewise Rigid Scene Model," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 1, pp. 1–28, 2015.

[135] C. Vogel, S. Roth, and K. Schindler, "View-Consistent 3D Scene Flow Estimation over Multiple Frames," in *European Conference on Computer Vision (ECCV)*, pp. 263–278, Springer, 2014.

[136] W. Luo, A. G. Schwing, and R. Urtasun, "Efficient deep learning for stereo matching," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 5695–5703, 2016.

[137] D. Ulyanov, A. Vedaldi, and V. S. Lempitsky, "Instance Normalization: The Missing Ingredient for Fast Stylization," *Computing Research Repository (CoRR)*, 2016.

[138] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[139] "PyTorch deep learning framework." http://http://pytorch.org/, accessed on May 5th, 2018.

[140] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond Empirical Risk Minimization," in *International Conference on Learning Representations (ICLR)*, 2018.

[141] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[142] L. A. Riggs, F. Ratliff, J. C. Cornsweet, and T. N. Cornsweet, "The disappearance of steadily fixated visual test objects," *Journal of the Optical Society of America*, vol. 43, p. 6, 1953.

[143] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, *et al.*, "A 640×480 dynamic vision sensor with a 9μm pixel and 300meps address-event representation," in *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 66–67, IEEE, 2017.

[144] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240×180 130 db 3μs latency global shutter spatio-temporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, 2014.

[145] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza, "Asynchronous, photometric feature tracking using events and frames," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 750–765, 2018.

[146] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza, "Event-based vision meets deep learning on steering prediction for self-driving cars," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018.

[147] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman, "HATS: histograms of averaged time surfaces for robust event-based object classification," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2018.

[148] J. Kogler, M. Humenberger, and C. Sulzbachner, "Event-based stereo matching approaches for frameless address event stereo data," in *International Symposium on Visual Computing*, pp. 674–685, Springer, 2011.

[149] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck, "Asynchronous event-based binocular stereo matching," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 23, no. 2, pp. 347–353, 2012.

[150] L. A. Camunas-Mesa, T. Serrano-Gotarredona, S. H. Ieng, R. B. Benosman, and B. Linares-Barranco, "On the use of orientation filters for 3D reconstruction in event-driven stereo vision," *Frontiers in Neuroscience*, vol. 8, p. 48, 2014.

[151] D. Zou, P. Guo, Q. Wang, X. Wang, G. Shao, F. Shi, J. Li, and P.-K. Park, “Context-aware event-driven stereo matching,” in *IEEE International Conference on Image Processing (ICIP)*, pp. 1076–1080, IEEE, 2016.

[152] D. Zou, F. Shi, W. Liu, J. Li, Q. Wang, P.-K. Park, and E. R. Hyunsurk, “Robust Dense Depth Map Estimation from Sparse DVS Stereos,” in *British Machine Vision Conference (BMVC)*, vol. 3, 2017.

[153] S. Schraml, A. Nabil Belbachir, and H. Bischof, “Event-driven stereo matching for real-time 3d panoramic vision,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2015.

[154] A. Zihao Zhu, Y. Chen, and K. Daniilidis, “Realtime time synchronized event-based stereo,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 433–447, 2018.

[155] E. Piatkowska, J. Kogler, N. Belbachir, and M. Gelautz, “Improved cooperative stereo matching for dynamic vision sensors with ground truth evaluation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 53–60, 2017.

[156] E. Piatkowska, A. Belbachir, and M. Gelautz, “Asynchronous stereo vision for event-driven dynamic stereo sensor using an adaptive cooperative approach,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops (ICCVW)*, pp. 45–50, 2013.

[157] M. Firouzi and J. Conradt, “Asynchronous event-based cooperative stereo matching using neuromorphic silicon retinas,” *Neural Processing Letters*, vol. 43, no. 2, pp. 311–326, 2016.

[158] Z. Xie, S. Chen, and G. Orchard, “Event-based Stereo Depth Estimation using Belief Propagation,” *Frontiers in Neuroscience*, vol. 11, p. 535, 2017.

[159] Z. Xie, J. Zhang, and P. Wang, “Event-based stereo matching using semiglobal matching,” *International Journal of Advanced Robotic Systems*, vol. 15, no. 1, 2018.

[160] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza, “Semi-dense 3D reconstruction with a stereo event camera,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 235–251, 2018.

[161] A. Andreopoulos, H. J. Kashyap, T. K. Nayak, A. Amir, and M. D. Flickner, “A low power, high throughput, fully event-based stereo system,” in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 7532–7542, 2018.

[162] G. Dikov, M. Firouzi, F. Röhrbein, J. Conradt, and C. Richter, “Spiking cooperative stereo-matching at 2 ms latency with neuromorphic hardware,” in *Conference on Biomimetic and Biohybrid Systems*, pp. 119–137, Springer, 2017.

[163] A. Nguyen, T.-T. Do, D. G. Caldwell, and N. G. Tsagarakis, "Real-Time 6DOF Pose Relocalization for Event Cameras with Stacked Spatial LSTM Networks," *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019.

[164] D. P. Moeys, F. Corradi, E. Kerr, P. Vance, G. Das, D. Neil, D. Kerr, and T. Delbrück, "Steering a predator robot using a mixed frame/event-driven convolutional neural network," in *2016 Second International Conference on Event-based Control, Communication, and Signal Processing*, pp. 1–8, IEEE, 2016.

[165] L. Wang, Y.-S. Ho, K.-J. Yoon, *et al.*, "Event-based High Dynamic Range Image and Very High Frame Rate Video Generation using Conditional Generative Adversarial Networks," *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, 2019.

[166] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis, "EV-FlowNet: self-supervised optical flow estimation for event-based cameras," *Robotics: Science and Systems*, 2018.

[167] C. Ye, A. Mitrokhin, C. Parameshwara, C. Fermüller, J. A. Yorke, and Y. Aloimonos, "Unsupervised Learning of Dense Optical Flow and Depth from Sparse Event Data," *Computing Research Repository (CoRR)*, 2018.

[168] D. Neil, M. Pfeiffer, and S.-C. Liu, "Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 3882–3890, 2016.

[169] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional LSTM network: A machine learning approach for precipitation nowcasting," in *Advances in Neural Information Processing Systems (NIPS)*, pp. 802–810, 2015.

[170] Y. Cao, Y. Chen, and D. Khosla, "Spiking deep convolutional neural networks for energy-efficient object recognition," *International Journal of Computer Vision (IJCV)*, vol. 113, no. 1, pp. 54–66, 2015.

[171] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci, "Event-based convolutional networks for object detection in neuromorphic cameras," *Computing Research Repository (CoRR)*, 2018.

[172] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman, "Hfirst: a temporal approach to object recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 37, no. 10, pp. 2028–2040, 2015.

[173] S. Tulyakov, A. Ivanov, and F. Fleuret, "Practical Deep Stereo (PDS): Toward applications-friendly deep stereo matching," in *Proceedings of the international conference on Neural Information Processing Systems (NeurIPS)*, pp. 5874–5884, 2018.

[174] S. Wang, S. Suo, W.-C. Ma, A. Pokrovsky, and R. Urtasun, "Deep parametric continuous convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2589–2597, 2018.

[175] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256, 2010.

[176] A. Z. Zhu, D. Thakur, T. Özaslan, B. Pfrommer, V. Kumar, and K. Daniilidis, "The Multi-Vehicle Stereo Event Camera Dataset: An event camera dataset for 3D perception," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 2032–2039, 2018.

[177] "Multi-Vehicle Stereo Event Camera Dataset (MVSEC)." https://daniilidis-group.github.io/mvsec/, accessed on March 9th, 2019.

[178] "Robotic Operation System (ROS)." http://www.ros.org/, accessed on March 9th, 2019.

[179] O. N. Thomas, G. Cremonese, M. Banaszkiewicz, J. Bridges, S. Byrne, V. Da Deppo, S. Debei, M. R. El-Maarry, E. Hauber, C. J. Hansen, A. Ivanov, W. Markiewicz, M. Massironi, A. S. Mcewen, C. Okubo, P. Orleanski, A. Pommerol, P. Wajer, and J. Wray, "The Color and Stereo Surface Imaging System (CaSSIS) for ESA's Trace Gas," in *Proceedings of Eighth International Conference on Mars*, vol. 1791, p. 1067, 2014.

[180] N. Thomas, G. Cremonese, R. Ziethe, M. Gerber, M. Brändli, G. Bruno, M. Erismann, L. Gambicorti, T. Gerber, K. Ghose, *et al.*, "The Colour and Stereo Surface Imaging System (CaSSIS) for the ExoMars Trace Gas Orbiter," *Space Science Reviews*, vol. 212, no. 3-4, pp. 1897–1944, 2017.

[181] N. Thomas, G. Cremonese, M. Almeida, M. Banaszkiewicz, P. Becerra, J. Bridges, S. Byrne, and V. Da Deppo, "CaSSIS-First Images from science orbit," in *European Planetary Science Congress*, vol. 12, 2018.

[182] N. Thomas, G. Cremonese, M. Almeida, M. Banaszkiewicz, J. Bapst, P. Becerra, J. Bridges, S. Byrne, S. Conway, V. da Deppo, *et al.*, "CaSSIS: Overview of Imaging in the First 9 Months of the Prime Mission," in *Lunar and Planetary Science Conference*, vol. 50, 2019.

[183] N. Thomas, G. Cremonese, M. Almeida, J. Backer, P. Becerra, G. Borrini, S. Byrne, M. Gruber, R. Heyd, A. Ivanov, *et al.*, "CaSSIS on the ExoMars Trace Gas Orbiter: Operational Approach," in *Lunar and Planetary Science Conference*, vol. 50, 2019.

[184] Z. Zhengyou, "Flexible camera calibration by viewing a plane from unknown orientations," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, vol. 1, pp. 666–673, 1999.

[185] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of the IEEE Computer Vision and Pattern Recognition (CVPR)*, pp. 1106–1112, 1997.

[186] R. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal on Robotics and Automation*, vol. 3, no. 4, pp. 323–344, 1987.

[187] “Single Camera Calibrator App.” https://ch.mathworks.com/help/vision/ug/single-camera-calibrator-app.html, accessed on May 23rd, 2017.

[188] “Camera calibration with OpenCV.” http://docs.opencv.org/2.4/doc/tutorials/calib3d/camera_calibration/camera_calibration.html, accessed on May 23rd, 2017.

[189] “Camera Calibration Toolbox for Matlab.” http://www.vision.caltech.edu/bouguetj/calib_doc/, accessed on May 5th, 2017.

[190] D. Claus and A. W. Fitzgibbon, “A rational function lens distortion model for general cameras,” in *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. 213–219, 2005.

[191] R. Grompone von Gioi, P. Monasse, J.-M. Morel, and Z. Tang, *Self-consistency and universality of camera lens distortion models.* Hyper Articles en Ligne (HAL), 2010.

[192] “VisieR the most complete library of published astronomical catalogues.” http://vizier.u-strasbg.fr/, accessed on June 2nd, 2010.

[193] M. Samaan, T. Griffth, P. Singla, and J. L. Junkins, “Autonomous on-orbit calibration of star trackers,” in *Proceedings of the IEEE Core Technologies for Space Systems Conference*, Nov 2001.

[194] M. Pal and M. S. Bhat, “Autonomous Star Camera Calibration and Spacecraft Attitude Determination,” *Journal of Intelligent & Robotic Systems*, vol. 79, no. 2, pp. 323–343, 2014.

[195] X. Junfeng, J. Wanshou, and G. Jianya, “On-orbit Stellar Camera Calibration Based on Space Resectioning,” in *Proceedings of the IEEE Congress of International Society for Photogrammetry and Remote Sensing Proceedings (ISPRS)*, vol. 37, 2005.

[196] A. Klaus, J. Bauer, K. Karner, P. Elbischger, R. Perko, and H. Bischof, “Camera calibration from a single night sky image,” in *Proceedings of the IEEE international conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, pp. I–I, 2004.

[197] D. Lang, D. W. Hogg, K. Mierle, M. Blanton, and S. Roweis, “Astrometry.net: Blind astrometric calibration of arbitrary astronomical images,” *The Astronomical Journal*, vol. 139, no. 5, p. 1782, 2010.

[198] “Astrometry.net.” http://astrometry.net/, accessed on May 24th, 2017.

[199] “NASA’s Observation Geometry System for Space Science Missions (SPICE).” https://naif.jpl.nasa.gov/naif/toolkit.html, accessed on May 24th, 2017.

[200] “USGS Integrated Software for Imagers and Spectrometers (ISIS).” https://isis.astrogeology.usgs.gov, accessed on June 6th, 2017.

[201] "ExoMars Trace Gas Orbiter (TGO) SPICE kernels." https://naif.jpl.nasa.gov/pub/naif/EXOMARS2016, accessed on May 24th, 2017.

[202] "NASA Ames Stereo Pipeline (ASP)." https://ti.arc.nasa.gov/tech/asr/groups/intelligent-robotics/ngt/stereo/, accessed on August 1st, 2019.

[203] "BAE Systems SOCET SET." https://www.geospatialexploitationproducts.com, accessed on August 1st, 2019.

[204] "S2P: Satellite stereo pipeline." https://github.com/MISS3D/s2p, accessed on August 1st, 2019.

[205] C. de Franchis, E. Meinhardt-Llopis, J. Michel, J.-M. Morel, and G. Facciolo, "An automatic and modular stereo pipeline for pushbroom images," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. II-3, pp. 49–56, 2014.

[206] E. Kilpelä, "Compensation of systematic errors of image and model coordinates," *Photogrammetria*, vol. 37, no. 1, pp. 15–44, 1981.

[207] "CDE: Lightweight application virtualization for Linux." http://www.pgbovine.net/cde.html, accessed August 1st, 2019.

[208] A. S. McEwen, E. M. Eliason, J. W. Bergstrom, N. T. Bridges, C. J. Hansen, W. A. Delamere, J. A. Grant, V. C. Gulick, K. E. Herkenhoff, L. Keszthelyi, *et al.*, "Mars Reconnaissance Orbiter's High Resolution Imaging Science Experiment (HiRISE)," *Journal of Geophysical Research: Planets*, vol. 112, no. E5, 2007.

[209] "Planetary Data System (PDS) archive." https://pds.nasa.gov/, accessed on August 1st, 2019.

[210] C. Re, S. Tulyakov, E. Simioni, T. Mudric, G. Cremonese, and N. Thomas, "Performance Evaluation of 3DPD, the Photogrammetric Pipeline for the Cassis Stereo Images," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4213, pp. 1443–1449, June 2019.

[211] P. Becerra, M. Sori, N. Thomas, S. Tulyakov, S. Sutton, A. Pommerol, E. Simioni, G. Cremonese, H. Team, and C. Team, "Climate Record Signals in the South Polar Cap of Mars from HiRISE and CaSSIS Stereo Imaging," in *Lunar and Planetary Science Conference*, vol. 50, 2019.

[212] P. Becerra, M. M. Sori, N. Thomas, A. Pommerol, E. Simioni, S. S. Sutton, S. Tulyakov, and G. Cremonese, "Timescales of the climate record in the south polar ice cap of Mars," *Geophysical Research Letters*, 2019.

[213] P. Becerra, M. Sori, N. Thomas, A. Pommerol, M. Almeida, S. Tulyakov, A. Ivanov, E. Simioni, and G. Cremonese, "Stereo-topographic mapping of the Stratigraphy of

Mars' South Polar Layered Deposits," in *European Planetary Science Congress*, vol. 12, 2018.

[214] A. Pommerol, N. Thomas, Z. Yoldi, V. Roloff, M. Almeida, P. Becerra, S. Tulyakov, L. Tornabene, F. Seelos, J. Bapst, *et al.*, "Ices, frosts and clouds on Mars observed by CaSSIS during the first months of TGO's primary science mission," in *European Planetary Science Congress*, vol. 12, 2018.

[215] S. Tulyakov, F. Fleuret, M. Kiefel, P. Gehler, and M. Hirsch, "Learning an event sequence embedding for event-based deep stereo," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019. To appear.

[216] R. Ye, F. Liu, and L. Zhang, "3D Depthwise Convolution: Reducing Model Parameters in 3D Vision Tasks," in *Canadian Conference on Artificial Intelligence*, pp. 186–199, Springer, 2019.

[217] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1251–1258, 2017.

[218] P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-Alone Self-Attention in Vision Models," *Computing Research Repository (CoRR)*, 2019.

[219] S. Woo, J. Park, J.-Y. Lee, and I. So Kweon, "CBAM: Convolutional Block Attention Module," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 3–19, 2018.

Stepan Tulyakov

Computer Vision | Machine Learning

1018 Lausanne, Switzerland
☎ *+41 78 640 71 31*
✉ *tulyakov.stepan@gmail.com*
stepan-tulyakov-a2324331
tlkvstepan

Education

2015–now **Ph.D., Computer Vision**, *École Polytechnique Fédérale de Lausanne*, Switzerland.
- Advisors Prof. François Fleuret and Dr. Anton Ivanov
- Expected graduation December 2019.

2008–2010 **M.S., Computer Vision**, *Seoul National University*, Korea.
- Funded by Samsung Electronics global talents program.
- Advisor Prof. Kyoung-Mu Lee
- GPA 3.98/4.3

2002–2008 **B.S. and M.S., Robotics**, *Bauman Moscow State University*, Russia.
- Advisor Prof. Ivan Rubcov
- GPA 4.87/5.0, Degree with honors

Work experience

2018–2019 **Machine Learning Intern**, *Amazon*, Tuebingen, Germany.

2014–2015 **Senior R&D Engineer**, *Samsung Electronics*, Suwon, South Korea.

2010–2014 **R&D Engineer**, *Samsung Electronics*, Suwon, South Korea.

2006–2008 **Software Engineer**, *Research Institute of Long Distance Communication*, Moscow, Russia.

Selected projects

Deep stereo reconstruction for an event camera
Proposed a novel method for an event sequence embedding and developed the first deep stereo reconstruction method for an event camera (*ICCV 2019, oral presentation*).

Applications-friendly deep stereo matching
Proposed a novel memory-efficient deep stereo network, leveraging large image context, less sensitive to ambiguous matches and applicable to any disparity range without re-training (*NeurIPS 2018, poster*).

Weakly supervised learning of deep similarity for stereo
Proposed a new method for learning deep similarity measure for stereo matching using task-specific constraints instead of a ground truth depth (*ICCV 2017, poster*).

Star-field calibration for a space telescope
Proposed a novel semi-automatic method for geometric calibration of a space telescope with complex optical distortions using multiple star-field images (*Advances in Space Research, 2018*) and applied it to Colour and Stereo Surface Imaging System on-board ESA's ExoMars Trace Gas Orbiter; developed tools for color mosaicing and stereo reconstruction.

Real-time zoom by fusion of wide-lens and tele-lens images
Proposed a novel real-time digital zoom method for devices with wide and tele cameras based on image fusion and implemented the method on a prototype Android device.

Interactive refocusing for a plenoptic camera
Proposed a novel method for multi-view stereo reconstruction (*ICIP 2013, oral*) for a plenoptic camera, developed calibration and refocusing algorithms and implemented them on a prototype Android device.

Real-time single image high dynamic range method for images and video Proposed a novel real-time single image high dynamic range method based on multi-scale adaptive histogram equalization for images and videos (*Bronze Award on Samsung conference, 2016*) and implemented the method on DSP for a photo camera and on FPGA for a digital TV.

Interactive texture segmentation
Developed a new random walk-based interactive segmentation method working with textures (M.S. thesis, 2010).

Multiple target tracking for a phased array radar
Implemented data association and multiple targets tracking algorithm for a phased array radar (M.S. thesis, 2008).

Technical skills

OS	Linux, MacOS, Windows
Languages	Python, Matlab, C; some experience with Lua, C++, CUDA.
Deep Learning	PyTorch, Torch7; some experience with MxNet.
Other	LaTeX, Git, OpenCV

Community service

Reviewer for NeurIPS, ICCV, CVPR, ICML, The Visual Computer Journal, TPAMI, Signal Processing Letters.

Languages

Russian (Native), English (Fluent)

Publications

Peer-reviewed articles

S. Tulyakov, F. Fleuret, M. Kiefel, P. Gehler, and M. Hirsch, "Learning an event sequence embedding for event-based deep stereo," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, Oral, to appear. [Online]. Available: https://fleuret.org/papers/tulyakov-et-al-iccv2019.pdf

S. Tulyakov, A. Ivanov, and F. Fleuret, "Practical Deep Stereo (PDS): Toward applications-friendly deep stereo matching," in *Proceedings of the international conference on Neural Information Processing Systems (NeurIPS)*, 2018, Poster. [Online]. Available: https://arxiv.org/pdf/1806.01677.pdf

S. Tulyakov, A. Ivanov, T. Nicolas, V. Roloff, A. Pommerol, G. Cremonese, T. Weigel, and F. Fleuret, "Geometric calibration of colour and stereo surface imaging system (cassis) of esa's trace gas orbiter," in *Advances In Space Research*, 2017. [Online]. Available: http://fleuret.org/papers/tulyakov-et-al-jasr2018.pdf

S. Tulyakov, A. Ivanov, and F. Fleuret, "Weakly supervised learning of deep metrics for stereo reconstruction," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017, Poster. [Online]. Available: http://fleuret.org/papers/tulyakov-et-al-iccv2017.pdf

S. Tulyakov, T. Lee, and H. Han, "Quadratic formulation of disparity estimation problem for light-field camera," *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2013. [Online]. Available: http://ieeexplore.ieee.org/document/6738425/

Patents

T.-H. Lee, S. Tuliakov, and H.-C. Han, "Photographing device and photographing method for taking picture by using a plurality of microlenses," Jul. 24 2014, US Patent App. 14/158,148.

T.-H. Lee, J.-G. Kim, S. Tuliakov, S.-H. Lee, S.-R. Jeon, K. Kyoung-Young, H.-S. Hong, and H.-C. Han, “Image photographing apparatus and image photographing method for generating a synthesis image from a plurality of images,” Nov. 7 2017, US Patent 9,813,615.

S. Tulyakov, T.-H. Lee, and H.-C. Han, “Image generating apparatus including digital iris and method and non-transitory recordable medium,” Jul. 12 2016, US Patent 9,392,187.

S. Tuliakov, T.-H. Lee, S.-K. Cho, and H.-C. Han, “Endoscope apparatus and control method for adjusting light beam based on geometry of body,” Aug. 30 2016, US Patent 9,427,174.