Van lichtstralen naar 3D-modellen

From Light Rays to 3D Models

Simon Donné

Promotoren: prof. dr. ir. W. Philips, prof. dr. ir. B. Goossens
Proefschrift ingediend tot het behalen van de graad van
Doctor in de ingenieurswetenschappen: computerwetenschappen

UNIVERSITEIT
GENT

Vakgroep Telecommunicatie en Informatieverwerking
Voorzitter: prof. dr. ir. H. Bruneel
Faculteit Ingenieurswetenschappen en Architectuur
Academiejaar 2017 - 2018

ISBN 978-94-6355-112-0
NUR 958, 950
Wettelijk depot: D/2018/10.500/30

Members of the jury: prof. dr. ir. Gert De Cooman (chair)
dr. ir. Jan Aelterman (secretary)
prof. dr. Peter Lambert
prof. dr. ir. Adrian Munteanu
Prof. Dr.-ing. Andreas Geiger
dr. ir. Hiep Luong
prof. dr. ir. Bart Goossens (promotor)
prof. dr. ir. Wilfried Philips (promotor)

The work contained herein was performed in the context of a doctoral scholarship by BOF (Bijzonder Onderzoeksfonds grant number 01D21213).

Acknowledgments

“ The purpose of a storyteller is not to tell you how to think, but to give you questions to think upon. ”

— Brandon Sanderson, *The Way of Kings*

It is needless to say that a work of this size and length has seen the support of many people, directly and indirectly; yet each and every one of them deserves my thanks. I would like to thank my supervisors, prof. Wilfried Philips and prof. Bart Goossens, for the opportunity to pursue my doctoral degree with them in the Image Processing and Interpretation group as well as for their support throughout the years. My (ex-)colleagues at Ghent University – whether I have collaborated with them or not – have made the past four years an enjoyable experience. I am as thankful for the many discussions about our research as I am about the well-timed distractions that may or may not have occurred during the occasional too-long lunch break. An especially honourable mention goes out to anyone that helped proofread the drafts for my papers or this manuscript. I am grateful for your sacrifices. I am sorry.

My family and friends, both at home and abroad, have been as supportive as I could have wished. They put up with deadline-me, and my subsequent disappearing to conferences and other trips, yet managed to get me back from both without fail.

And yes, I will finally get a “real job” now.

Ghent, March 27th, 2018
Simon Donné

Table of Contents

List of Figures

List of Tables

Nederlandse samenvatting

Een van de meest fundamentele problemen in computer visie is beeldgebaseerde 3D reconstructie. Op basis van verschillende beelden van camera's die de scène waarnemen, wensen we haar 3D structuur te schatten. Dit heeft vele mogelijke toepassingen: misschien willen we het 3D model van de scène zodat we er mee kunnen interageren of erdoor kunnen navigeren (zoals het geval is bij robotvisie of autonome voertuigen), omdat we de scène vanuit nieuwe gezichtspunten willen weergeven (denk bijvoorbeeld aan virtuele avatars in een videoconferentie), of omdat we niet-triviale 3D metingen willen uitvoeren (bijvoorbeeld de automatische lengte- en oppervlaktemetingen van planten).

In dit werk bespreken we de volledige workflow van 3D reconstructie, van de opgenomen beelden tot het gereconstrueerde model. We starten met een geparametriseerd wiskundig model van de camera's, en schatten hiervan dan de parameters: hun positie, oriëntatie en projectiemodel. Met deze kennis over de camera's pogen we dan de 3D-naar-2D projectiestap van de camera's te inverteren. Dat dit überhaupt mogelijk is, getuigen onze ogen: we zijn in staat ons een nauwkeurig beeld te vormen van de 3D wereld rondom ons op basis van slechts twee 2D beelden.

In eerste instantie bespreken we het wiskundige model van de verschillende camera's die de scène waarnemen, en de parameters die hun gedrag beschrijven. Deze camera's kunnen standaard kleurencamera's zijn, maar we bespreken ook alternatieven zoals thermische infraroodcamera's, die een probleem gevormd hebben op het vlak van calibratieobjecten, en hyperspectraalcamera's, die niet alleen een veel grotere waaier aan golflengtes opmeten, maar ook een andere projectietechniek gebruiken. In het bijzonder hebben we voor de laatste een robuustere calibratiemethode ontwikkeld, aangezien de toenmalige state-of-the-art numeriek instabiel werd in veel-praktisch voorkomende situaties.

Om de cameraparameters van deze wiskundige modellen te schatten steunen we op zogenoemde calibratieobjecten, die ons overeenkomsten geven tussen gekende 3D punten en 2D waarnemingen ervan. Die objecten worden typisch gedetecteerd aan de hand van aparte algoritmes die specifiek ontwikkeld werden voor elk object apart, wat een significante onderzoeksinspanning vergt. We hebben een grote waaier aan calibratieobjecten te onzer beschikking. Alhoewel veruit de meest gebruikte variant het 2D schaakbord is, is het niet de enige mogelijkheid, en is het in bepaalde instanties niet bruikbaar. Bij thermische camera's, bijvoorbeeld, kunnen we geen geprinte 2D patronen gebruiken, omdat die niet genoeg contrast geven in de thermische modaliteit. Om de extra onderzoeksinspanning voor het gebruik van nieuwe soorten calibratieobjecten in te dijken, hebben we het gebruik

van deep learning voor calibratieobjectdetectie voorgesteld. Een dergelijke techniek kan eenvoudig opnieuw getraind worden om een nieuw soort calibratieobject te detecteren, en vergt daarbij maar een kleine hoeveelheid grondwaarheid. Op die manier moeten we geen nieuw detectie algoritme ontwikkelen wanneer we gedwongen worden een nieuw calibratiepatroon te gebruiken.

Eens we de camera's in de scène nauwkeurig gemodelleerd hebben, gaan we over tot de 2D-naar-3D reconstructie. Dit kan gaan van de plaatsbepaling van een enkel punt (triangulatie), over de reconstructie van een puntenwolk, tot een diepeschatting voor elke pixel in de inputafbeeldingen.

Het meest fundamentele probleem is punttriangulatie: op basis van de 2D metingen van een aantal camera's willen we de meest waarschijnlijke achterliggende 3D locatie vinden. De meeste bestaande technieken minimaliseren hiervoor de ℓ_2-norm, maar die heeft problemen met lokale minima, en om het globale optimum betrouwbaar te vinden moeten we ons tot complexe en trage methodes keren. Als alternatief lijdt ℓ_∞-norm triangulatie niet onder dergelijke lokale minima, maar de bestaande methodes hier zijn traag en complex (en moeilijk te implementeren op een GPU). Om dit probleem te behandelen hebben we een nieuw algoritme ontwikkeld dat ℓ_∞-norm triangulatie sneller oplost dan bestaande methodes, en dat bovendien zonder veel problemen op een GPU geïmplementeerd werd.

Uiteraard willen we uiteindelijke objecten in hun geheel reconstrueren. In de praktijk zijn voorwerpen sterk beperkt in hun vervormingen (als ze al niet volledig rigide zijn). Als we alle punten onafhankelijk trianguleren, gooien we die samenhang weg. In tegenstelling stellen non-rigid structure-from-motion (NRSFM) technieken de punten van een object voor in een laag-dimensionale ruimte om zo het aantal te schatten parameters te verlagen. Dit heeft onder meer voor gezichtsreconstructie zijn waarde al bewezen: een gezicht kan goed voorgesteld worden in een dergelijke laag-dimensionale ruimte. Helaas zijn bestaande NRSFM technieken niet opgewassen tegen het on-line verwerken van videostromen, zoals het geval is bij teleconferencing scenario's. Om dit op te lossen hebben we een elegant algoritme ontwikkeld dat de volledige geschiedenis van geziene beelden samenvat in een laag aantal keyframes. Op die manier kunnen we er wel in slagen om de gezichten in een videoconferentie te reconstrueren.

Maar zelfs een dergelijke reconstructie is nog altijd gebaseerd op een relatief laag aantal punten. Voor sommige toepassingen, zoals de nauwkeurige modelleren van groeiende planten, willen we een gedetailleerdere 3D reconstructie. Multiview stereopsis technieken bieden hier soelaas door de afstand tot iedere pixel te schatten. Op die manier kunnen ze redundanties en de samenhang in de resulterende dieptemappen uitbuiten, maar zijn ze – door het grote aantal onbekenden dat geschat dient te worden – vaak traag. Daarom onderzochten we het gebruik van oversegmentatie voor efficiëntere stereo matching. Naast dit tijdsaspect, vormen ook occlusies een probleem – in het bijzonder voor binoculaire stereo matching. Door een uitbreiding van de traditionele stereo matching te formuleren die dieptemappen schat gebaseerd op basis van meerdere beelden, konden we de uitgebreide literatuur over occlusiedetectie in de binoculaire setting uitbuiten.

We eindigen met een bespreking van een praktische toepassing die gebaat is bij 3D reconstructie: het uitvoeren van automatische metingen op maisplanten. In samenwerking met onderzoekers van het Vlaams Instituut voor Biotechnologie, aan de groep Plant Systems Biology, hebben we met PhenoVision gewerkt: een geautomatiseerde serre die een groot aantal planten kan verwerken, en ervan beelden kan capteren. Om de functionaliteit uit te breiden met automatische metingen (zoals bijvoorbeeld bladlengte, biomassa of groeisnelheid), hebben we need aan verscheidene van de technieken die in dit werk aan bod komen: we dienen traditionele kleurencamera's te calibreren, maar ook twee hyperspectraalcamera's en een infrarood sensor. Uiteindelijk leidt dit tot 3D reconstructie van de planten, met daaropvolgend een geometrische modellering in de vorm van een set curves in de ruimte waarvan we eenvoudig de lengte kunnen berekenen.

English summary

One of the prime challenges in computer vision is image-based 3D reconstruction. Based on several images by cameras observing the scene, we wish to estimate its 3D structure. This has numerous applications: we might want the 3D model in order to interact with it or navigate through it (as is the case for robotic vision or autonomous vehicles), because we want to virtually view the scene from different vantage points (think personal avatars in a virtual 3D world for teleconferencing), or because we wish to perform non-trivial 3D measurements of the scene (for example, the automatic length and area measurement of plants throughout their development).

In this work we cover the end-to-end workflow of 3D reconstruction, from input images to the output 3D model. The pipeline starts with a parametrized mathematical modeling of the cameras in the scene, and subsequently the estimation of those parameters. Using this knowledge about the camera's position, orientation and imaging process, we then aim to invert to 3D-to-2D projection of the cameras. That this is possible to start with is evidenced by our own eyes: based on just two 2D views of the world, we are accurately able to gauge the 3D structure of the world.

First, we discuss the mathematical model of the various cameras viewing a scene and the parameters that dictate their behavior. These cameras include regular color cameras, but we also discuss alternatives such as thermal infrared cameras, which pose a challenge for calibration, or hyperspectral cameras, which not only capture far more wavelengths but also use a non-standard projection when imaging the scene, different from hand-held consumer cameras. Specifically for the latter, we have proposed a more robust calibration technique, as the hitherto state-of-the-art faced numerical instability in many practical situations.

When estimating the camera parameters of these mathematical models, we use so-called calibration objects, to provide us with correspondences between known 3D locations and their 2D observations. Calibration objects are typically detected in images using separate algorithms specifically designed for each calibration object, which requires significant research. A wide variety of calibration objects exist. Although the most widely used type is by far the 2D checkerboard pattern, they are not the only one, and they are not usable in all cases. For example, thermal cameras preclude the use of printed 2D patterns because they do not have any contrast in their modality. We have proposed the use of deep learning to calibration object detection, which can be retrained to detect different patterns with a small amount of ground truth annotation. In this way, there is no need to manually de-

sign a novel detection algorithm for a new calibration pattern when the existing ones do not suffice.

Using this information on the cameras in the scene, we take a look at various alternatives for 2D-to-3D reconstruction. This ranges from single-point reconstruction, over point cloud reconstruction to the dense depth estimation of all pixels in the input images.

The most simple problem is that of single point triangulation: based on measurements by several cameras, we wish to find the most likely 3D point to have caused those measurements. Most existing techniques use ℓ_2 norm minimization to find the most likely point. Yet the ℓ_2 has issues with local minima, and to reliably find the global optimum we require complex and slow techniques. On the other hand, ℓ_∞ norm triangulation does not suffer from local minima but existing methods are slow as well as complex (impeding their implementation on a GPU). To address this, we have developed a triangulation approach that solves ℓ_∞ point triangulation faster than established approaches, as well as being straightforward to implement on GPU for massive parallelization.

Naturally, we eventually wish to reconstruct entire objects. In practice, objects are restricted in their deformations (if they deform at all). By triangulating all points independently, we discard this temporal information. While point triangulation retains its value as a fast basic reconstruction technique, in the case of deforming objects, non-rigid structure-from-motion (NRSFM) techniques model an object's constituent points in a low-dimensional manifold in order to lower the number of parameters that need to be estimated (without which, the problem would be ill-posed). This has proven successful, among others, for face reconstruction: the face is well represented by a relatively low number of points that deform in an even lower-dimensional space. However, existing NRSFM techniques were not intended for on-line processing of video streams such as we encounter in a teleconferencing setting. We have proposed an elegant keyframe selection algorithm that condenses all of the history into a low number of selected keyframes, and in doing so enables the efficient on-line processing required by teleconferencing.

As we wish to reconstruct a scene with more and more points, comprising multiple objects that deform independently, modeling them in a low-dimensional manifold gets harder and harder. Yet for some applications, such as the accurate modeling of growing plants, such a dense reconstruction is necessary and the above sparse techniques don't cut it. Instead, multi-view stereopsis techniques estimate the depth of each pixel in the input image to arrive at a very dense point cloud reconstruction; by doing this, redundancy and coherency in the resulting depth maps can be exploited to alleviate the ill-posedness. Sadly, such methods are – by vice of the large number of unknowns to estimate – often complex and slow. We have successfully investigated the use of image over-segmentation for more efficient, and highly parallelizable, stereo matching techniques. Additionally, occlusions complicate the problem, especially in the two-view case. We have extended the traditional binocular stereo matching approach to multiple images. In this way, we can leverage the extensive literature on occlusion detection and mitigation from binocular stereo matching on the multi-view case.

Finally, we discuss a concrete scenario that requires 3D reconstruction: automatic maize plant measurements. In cooperation with biotechnical researchers at the Flemish Institute for Biotechnology, at the group of Plant Systems Biology, we have worked with PhenoVision: a greenhouse-like growth environment that can automatically tend to and image many plants. In order to extend its functionality to automatic measurements (such as leaf length, biomass or growth speed), we have needed many of the techniques discussed in this work: from camera calibration of both RGB and hyperspectral cameras, to 3D reconstruction and subsequent geometrical modeling.

1 Introduction

" In the beginning there was nothing, which exploded. "

— Terry Pratchett, *Lords and Ladies*

In order to effectively interpret the world around them, and to interact with it, computer systems often require knowledge of its 3D structure. However, replicating our own understanding of the world, and explicitly estimating its 3D structure, is a major challenge. It is often addressed by observing the scene with multiple cameras, and by subsequently trying to invert the 3D-to-2D projection that observation entails. While it may appear that 3D information is lost during this projection, and while it may not be intuitively obvious that it is possible to re-extract this information from the 2D images, this is indeed possible. This fact is proven by our own eyes and the so-called parallax effect: by closing the left and right eyes alternately, objects shift to the left or the right, with the shift depending on the object's distance to us. This is exactly the effect that many computer vision algorithms exploit to extract 3D information from 2D images.

Information about the 3D structure of the scene is valuable in a wide range of applications: from the topical autonomous navigation (as is the case for robotic vision or autonomous vehicles), over generating novel viewpoints of the scene (specifically, we discuss personal avatars for teleconferencing later on), to performing semantical analysis and non-trivial measurements on the resulting 3D model (such as we exemplify with the maize plant measurements in the BAHAMAS project). Reasoning on a 3D representation of the scene is often much easier than directly doing so on the 2D observations.

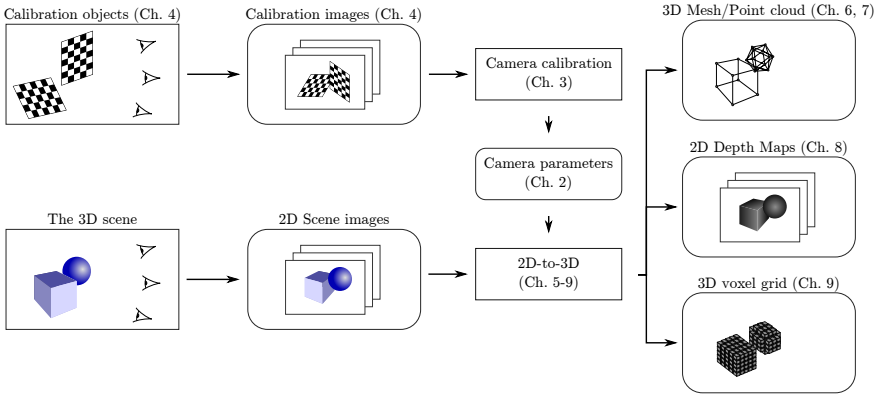


Figure 1.1: We identify two distinct tasks in the 3D reconstruction workflow: camera calibration to estimate the camera parameters based on calibration images and 2D-to-3D scene reconstruction to estimate the 3D structure based on scene observations and on these parameters.

In this thesis, we discuss the entire 3D reconstruction process in computer vision, which is shown schematically in figure 1.1. We identify two main blocks: first, the camera calibration to accurately model the cameras observing the scene, and next the 2D-to-3D reconstruction step that estimates the 3D structure from the input images. Because of the ambiguity between (1) the complete pipeline of 3D reconstruction as illustrated in the figure, i.e. including camera calibration, and (2) the specific approaches for converting 2D images to a 3D model, we will refer to the latter as 2D-to-3D reconstruction whenever confusion might arise.

The first step to inverting the projection is the camera calibration: obtaining a detailed model of the way the cameras observe the 3D scene as a 2D image. The imaging process can be described with an explicitly parametrized mathematical projection model, where the camera parameters contain the projection model parameters as well as its location and orientation in the 3D world. These camera parameters can be estimated either from specifically captured calibration images, which link 2D observations with known 3D points, or from the scene itself. While the calibration of the traditional perspective camera model is well understood and considered solved in literature, calibrating novel types of cameras still requires more research towards robust calibration procedures. One such example is hyperspectral cameras, which are often constructed as linear pushbroom cameras rather than perspective cameras. In the previously mentioned automatic analysis of maize plants, we needed to estimate the parameters of such a camera; unfortunately, the existing method for calibration had issues with our set-up (which was non-adjustable), as the camera poses resulted in numerical instabilities. We have

contributed a more robust calibration technique that addresses these issues.

In order to provide the link between 2D observations and known 3D points, calibration objects are used: objects whose structure is well known, and which are distinctly different from the scene (so that it is possible to automatically detect them). By far the most widely used is the 2D checkerboard pattern, but it is not applicable in all scenarios: sometimes the checkerboard is not easily usable, such as with thermal cameras where a simple printed object is not easily visible and it is harder to construct a checkerboard pattern with thermal contrast than it is to create, e.g., the perforated metal plate we used for the BAHAMAS project. However, reliably detecting the calibration objects and locating their feature points is not straightforward, and traditionally done by algorithms that need to be manually designed for each different type of calibration object. To address this inflexibility, we have developed a deep-learning based detection method that is straightforward to adapt to new varieties of calibration objects, as it only needs to be retrained with different supervision, rather than redesigned. At the same time, it matches or improves the performance of existing approaches on checkerboard detection.

Having obtained an accurate model of the 3D-to-2D projection, we then reconstruct the world in 3D by relating the measurements from different vantage points to each other: per the parallax effect, the positions at which the cameras observe a point are linked to its 3D location. Depending on the application, the estimated 3D structure can be represented in different ways: reconstructing the scene as a small set of points, as a mesh, as a set of depth maps, or as an occupancy grid (for each sample point on the occupancy grid, it expresses whether it is occupied or not). For example, in the teleconferencing setting it has proven sufficient to model the face as a small number of points, but autonomous navigation or plant modeling is much easier with the dense representation of an occupancy grid.

We discern two main groups of techniques in this thesis: sparse 2D-to-3D reconstruction methods, and dense techniques such as stereo matching or general multi-view stereopsis. Sparse methods focus on estimating the location of a small set of salient points. They are relatively lightweight and fast but only result in a sparse representation of the scene topology. Stereo matching techniques mimic the functionality of the human visual system: based on two closely spaced cameras (i.e. the eyes), they infer depth maps for the input views in order to represent the 3D structure of the scene. However, due to the dense processing of estimating depth for every pixel, stereo matching techniques are computation-heavy. Additionally, binocular stereo matching techniques are restricted to viewing the scene from only one side, and have issues with problems such as occlusions. General multi-view reconstruction methods combine an unrestricted set of cameras into either a set of depth maps or a full 3D representation in the form of an occupancy grid or dense meshes. The challenge here is to do this efficiently, and to correctly relate measurements from wildly differing viewpoints.

We now outline the contents of this dissertation and give a detailed overview of its contributions to the field of computer vision.

1.1 Outline

We start off with a primer on camera models in chapter 2 and model the projection during the image acquisition process mathematically for several camera models, from the tradition perspective pin-hole camera to more complex ones such as linear pushbroom cameras and plenoptic cameras. We then discuss the estimation of the parameters in these models in chapter 3. While the traditional perspective and orthographic camera models are well studied, this is less the case for other, rarer, camera models. Specifically, we discuss the calibration of linear-pushbroom (LPB) cameras, which are often used in hyperspectral camera systems. We show how the existing calibration technique for LPB cameras is numerically unstable in common situations, and discuss our proposed solution [Donné 2017b].

In order to perform camera calibration, we require a set of 2D observations that are linked to known set of 3D points. Typically we achieve this with calibration objects which have a well-known 3D structure, but these need to be detected and located in the input images to yield the 2D observations. As we discuss in chapter 4, a wide range of calibration objects are available, each best suited for different scenarios. Detecting the objects in the images is possible, as they are designed to be distinctly different from the scene, but existing techniques heavily rely on painstakingly designed detection algorithms that can only detect one specific type of calibration object – using a different one requires the design of a novel detection method. To address this, we discuss a new deep learning solution that is flexible enough to be easily retrained for different calibration objects [Donné 2016b].

Having obtained calibrating the cameras, we head on to the actual 2D-to-3D reconstruction step. We start with an overview of the entire field of 2D-to-3D reconstruction and the various approaches available, in chapter 5. These can generally be split up in two groups: techniques exploiting sparse point sets, which estimate the 3D location for only a subset of the points in the scene, and techniques for dense point sets, which estimate either a depth value for all of the input pixels or a dense 3D grid of information, e.g., occupancy.

In chapter 6 we discuss the efficient triangulation of single points of interest based on observations by several cameras. While most existing techniques use ℓ_2-norm minimization, this can encounter issues with local minima that require complex and costly techniques to overcome. The ℓ_∞ norm, an alternative, does not suffer from this drawback, but existing techniques for minimizing it are also slow and complex; we outline our rigorous approach based on geometric reasoning for ℓ_∞ point triangulation [Donné 2015c], which is more efficient and faster than pre-existing techniques and is straightforward to implement on GP-GPU.

As an alternative to single-point triangulation, we can also reconstruct larger groups of feature points on the same object as a whole, which lets us more naturally model deforming objects. So far, we have ignored the temporal aspect, and indeed for the most part we only discuss the reconstruction of rigid scenes. However, when the scene changes from frame to frame, the basic problem becomes ill-posed: for each $3D$ point, we have 2 measurements (X and Y) for each of the C views, but as its location might change, there are $3C$ unknowns: there are less observations than we have unknowns! On the other hand, the deformation of objects is strongly restricted, and this allows non-rigid structure-from-motion (NRSFM) techniques to assume that the point clouds reside in a low-dimensional manifold, resulting in a well-posed problem. However, pre-existing NRSFM techniques act on prerecorded videos as whole, and cannot be used in an on-line processing mode, in which video is processed as it comes in, rather than "after the video has finished". In teleconferencing, such an on-line processing mode is indispensable. We propose an efficient keyframe selection method in chapter 7 to condense the possibly very large set of past frames without losing much information, so that it becomes feasible to use NRSFM in an on-line setting [Donné 2014].

In the context of dense reconstruction, we examine stereo matching techniques in more detail. These techniques estimate a depth value for every pixel in the input images based on two cameras observing the scene. Stereo matching is rather computationally expensive, an aspect which is only compounded by the constant rise in image resolution. To alleviate this we present the application of superpixels to accelerate stereo matching techniques on high resolution images in chapter 8 [Donné 2015a]. Secondly, we discuss the extension of binocular stereo matching techniques to use multiple images while still restricting the camera set-up to the same rectified setting as binocular stereo matching [Donné 2015b, Donné 2017a]. Doing helps mitigate the occlusion problem that binocular stereo matching techniques face.

We finish with an overview of the BAHAMAS project in chapter 9, where we have combined the above techniques for performing automated measurements on maize plants [Donné 2016d, Donné 2016e]. It utilizes several of our proposed improvements and techniques, illustrating how the various steps discussed above come together in a practical computer vision project.

1.2 Contributions and publications

Below, we list the publications resulting from this doctoral work. In addition to these, the research was also valuable in a variety of collaborations and industrial projects:

- The on-line NRSFM technique in chapter 7 was partly created during an internship at the Alcatel-Lucent Bell Labs research center in Antwerp.

- The work on efficient stereo matching and multi-image stereo matching from chapter 8 was used in the context of the ASPRO+ ICON project, which aimed to create high-quality 3D content for a goggle-free 3D cinema.
- As discussed in chapter 9, the work on automated measurements of maize plants was done as part of the BAHAMAS ICON project, a collaboration between the Flemish Institute of Biotechnology (VIB) and the Image Processing and Interpretation (IPI) research group.

1.2.1 Publications in international journals

Over the course of the doctoral work, two papers were published as first author in international, peer-reviewed journals:

- *MATE: Machine Learning for Adaptive Calibration Template Detection*, Simon Donné, Jonas De Vylder, Bart Goossens and Wilfried Philips. MDPI Sensors, 2016 [Donné 2016b].
- *Line-constrained camera location estimation in multi-image stereo matching*, Simon Donné, Bart Goossens and Wilfried Philips. MDPI Sensors, 2017 [Donné 2017a].

1.2.2 Publications in international, peer-reviewed conferences

I have presented 7 publications as the first author at international, peer-reviewed conferences for dissemination by the scientific community:

- *Online non-rigid structure-from-motion based on a keyframe representation of history*, Simon Donné, Ljubomir Jovanov, Bart Goossens, Wilfried Philips, Aleksandra Pižurica. International Conference on Computer vision Theory and Applications, VISAPP 2014 [Donné 2014].
- *Fast and robust variational optical flow for high-resolution images using SLIC superpixels*, Simon Donné, Jan Aelterman, Bart Goossens and Wilfried Philips. Advanced Concepts for Intelligent Vision Systems, ACIVS 2015 [Donné 2015a].
- *Variational multi-image stereo matching*, Simon Donné, Bart Goossens, Jan Aelterman and Wilfried Philips. International Conference on Image Processing, ICIP 2015 [Donné 2015b].
- *Point triangulation through polyhedron collapse using the ℓ_∞ norm*, Simon Donné, Bart Goossens, Wilfried Philips. International Conference on Computer Vision, ICCV 2015 [Donné 2015c].

- *Machine Learning for Maize Plant Segmentation*, Simon Donné, Hiep Luong, Bart Goossens, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé and Wilfried Philips. The Belgian-Dutch Conference on Machine Learning, BENELEARN 2016 [Donné 2016e].
- *Exploiting Reflectional and Rotational Invariance in Single Image Superresolution*, Simon Donné, Laurens Meeus, Hiep Quang Luong, Bart Goossens and Wilfried Philips. International Conference on Computer Vision and Pattern Recognition, CVPR 2017 [Donné 2017c].
- *Robust Plane-based Calibration for linear cameras*, Simon Donné, Hiep Luong, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé, Bart Goossens and Wilfried Philips. Proceedings of the International Conference on Image Processing, ICIP 2017 [Donné 2017b].

Of these, only our paper on single image superresolution [Donné 2017c] is not elaborated on in this dissertation, as it falls outside of the scope of 3D reconstruction.

1.2.3 Other publications

Some of the work contained herein was presented for dissemination at local conferences, or other events, without formally being published:

- *GPU-based maize plant analysis: accelerating CNN segmentation and voxel carving*, Simon Donné, Bart Goossens, Stijn Dhondt, Nathalie Wuyts, Hiep Luong, Dirk Inzé and Wilfried Philips. NVIDIA European GPU Technology Conference, NVIDIA EGTC 2016 [Donné 2016c].
- *3D Reconstruction of maize plants in the PhenoVision system*, Simon Donné, Hiep Luong, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé and Wilfried Philips. Knowledge for Growth, KfG 2016 [Donné 2016d].
- *Efficiently simulating n-body systems*, Simon Donné. HiPEAC Fall Computing Week 2016 Competition [Donné 2016a]

Of these, only the publication on efficiently simulating n-body systems is not discussed in this dissertation [Donné 2016a]. It outlines an implementation of efficient physics simulations of very large point clouds, by splitting the load over multiple GPUs, rather than a computer vision problem, and does not fit well within the scope of this manuscript.

2
Camera acquisition models

" The dinosaurs didn't have a space program, much less telescopes, so it didn't end very well for them. "

— Alessondra Springmann*, on deflecting Earth-bound asteroids.*

The first step in interacting meaningfully with the world around us, is observing it. Modeled after the best working machines we know, our own brains, computer vision systems use cameras to observe the environment. These cameras project the 3D world onto a 2D representation, a projection which we later wish to invert. To do so, we first need to model this 2D projection accurately. This chapter introduces the various aspects of measuring a scene as a digital image: we express mathematically how cameras project a 3D point onto their 2D image planes, and discuss other aspects such as the modalities (the set of wavelengths) the cameras observe and image noise.

In the first section, we discuss the way cameras capture light onto their image plane, outlining and mathematically modeling several types of projections. We discuss four types of camera, as shown in figure 2.1: perspective cameras, orthographic cameras, linear pushbroom cameras, and plenoptic cameras. First of all, we discuss the perspective model, with the how and why of the requirements for lenses; the perspective model is a good fit for most consumer cameras, and by far the most prevalent model in computer vision.

Additionally, we discuss three less common camera models. The orthographic camera model, as used by telecentric cameras, can serve as an approximation for the perspective camera model with a simpler mathematical formulation (known

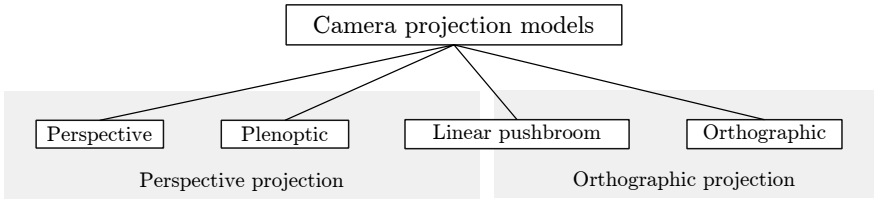


Figure 2.1: The different camera projection models. A plenoptic camera is the collection of many perspective cameras in one system, while a linear pushbroom camera is perspective in one direction, but orthographic in the other.

as the weak perspective projection) – it is used as such in chapter 7 to simplify the 2D-from-3D reconstruction problem in Structure-from-Motion. As the result of taking multiple measurements by sweeping a perspective line sensor over the scene (rather than a single observation by a 2D grid of sensors), a linear pushbroom (LPB) camera is modeled as a mix of the orthogonal and perspective camera model; it is used in many modern hyperspectral cameras. Plenoptic cameras image the scene from many slightly differing viewpoints, yielding a large set of different vantage points for subsequent 2D-to-3D reconstruction.

The last part of this chapter discusses three practical aspects of capturing images for subsequent image processing and computer vision: the discrete nature of digital images, imaging modalities, and image noise. First of all, the image plane consists of a discrete sampling grid of pixel sensors, but the preceding mathematical discussion is agnostic of discretization. Therefore, we briefly outline the effects of this discretization and the resulting practical limitations. Secondly, different cameras can capture a variety of different modalities and not just the visual spectrum of wavelengths our own eyes perceive. We discuss the modalities that we have worked with in this thesis: short-wave and thermal infrared, as well as the principle of a hyperspectral camera, which captures many wavelengths at once. Finally, we end with a short discussion on the source and mathematical model of imaging noise.

2.1 Projection models

Through projection of a 3D scene onto a 2D image, information is lost: information along the depth axis of the camera. However, depending on the type of projection, the depth of objects is encoded into their projection. We distinguish two categories of projections: perspective and parallel. In both cases, an image plane of sensors measures the appearance of an object by the light that strikes its photosensitive pixels as illustrated in figure 2.2. Whereas under a perspective projection the rays that are measured by the image plane all intersect in a single (focal)

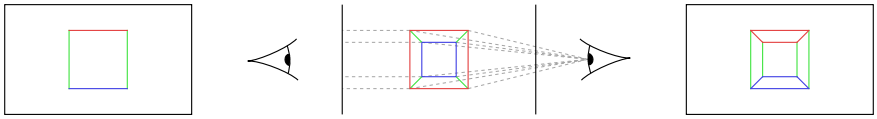


Figure 2.2: Orthographic projection (left) and perspective projection (right) of a 3D cube (middle). For orthographic projection, which is a type of parallel projection, the measured light rays are all parallel and orthogonal to the image plane. The left part of the figure shows that a cube whose sides happen to be parallel with the projection direction is imaged as a square. The right part of the figure shows that for perspective projection the rays all pass through the camera location, and imaging that same cube shows the back panel as being smaller than the front panel.

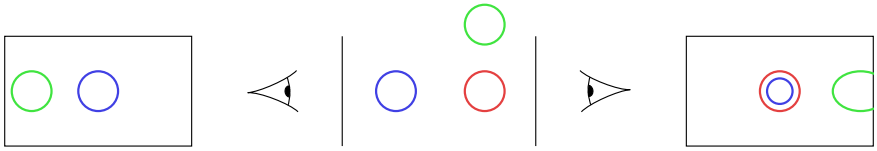


Figure 2.3: The effect of a sphere's position on its projection. For orthographic projection, the distance of the sphere does not make a difference and the two bottom spheres have an identical projection, but under perspective projection, the farther sphere appears smaller. A translation parallel to the image plane (top sphere) only results in the same translation in the orthographic projection, but also causes a skew in the perspective case (resulting in an ellipse).

point, a camera with a parallel projection model only measures light rays parallel to a given direction.

The main difference between both types of projection is the change in the projection due to imaging an object at different locations. For parallel projections, movement along the direction of the projection rays leaves the projection untouched, while movement parallel to the ray direction results in exactly the same translation of the projection on the image plane.

The only type of parallel projection we consider is the orthographic projection, in which the rays are orthogonal to the image plane. Under a perspective projection, an object moving away from the camera results in a smaller projection on the image plane, while movement parallel to the image plane will skew the projection as well as translating it. Both behaviours are illustrated in figure 2.3.

We start by discussing the perspective camera model in detail. The most simple form of a perspective camera is the so-called camera obscura, which images the scene by using a very small aperture: only light rays passing through the aperture (the focal point) can reach the sensors. In practice, such a small aperture means that there is not much light reaching the sensor, and the resulting image is very dim.

On the other hand, larger apertures result in blurrier images. To mitigate this

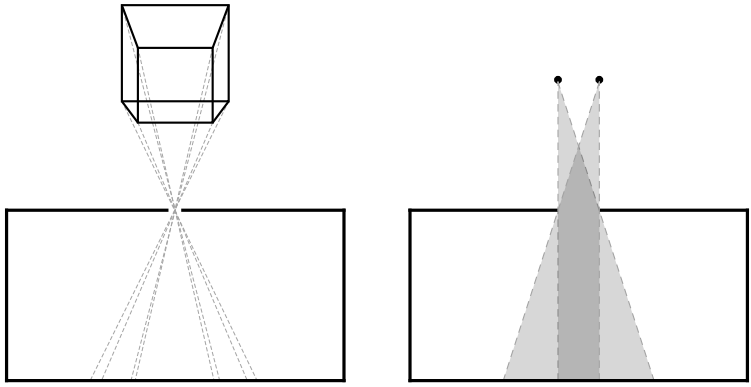


Figure 2.4: Left: a camera obscura, also known as a pinhole camera. The only light rays reaching the imaging film are the ones passing through the infinitely small hole. As can be seen in the figure the image is point-inverted around the aperture, so in practice we will invert the captured image again. This is equivalent to a virtual image plane in front of the aperture rather than behind it. Right: pin-hole camera with hole of finite size. The image is blurred as now a cone of light rather than an infinitely thin ray reaches the imaging plane at the end of the camera obscura.

trade-off, lenses are introduced to the camera system. A perfect lens focuses the light from the scene, resulting in more photons striking the sensor without causing such bad blur as does a larger aperture.

After we mathematically formulate the projection procedure of a perspective camera, we do the same for the orthographic camera model (which is used as an approximation for the perspective camera in chapter 7), the linear pushbroom camera (often used in hyperspectral cameras) and the plenoptic camera (which captures the scene from many slightly different vantage points).

2.1.1 The camera obscura: an ideal perspective camera

The most basic model of a perspective camera is the so-called camera obscura, also known as the pinhole camera [Lord Rayleigh 1891]. It consists of a small hole, the aperture, through which all light rays pass before striking the image plane: this is the focal point of this type of camera. As seen in figure 2.4 this results in a point-inversion of the imaged object. When viewing the images, they are typically inverted again, which is equivalent to having the image plane in front of the camera location, as we have shown it in figure 2.2 and figure 2.3.

As the image is formed by light rays passing through a tiny hole, next to no light strikes the image plane, which means that the resulting image is either very dim, or very noisy after amplifying the signal. This is due to the discreteness of light photons, which causes sampling noise as discussed in section 2.4.

Figure 2.5: Examples of motion blur: the imaging sensors are exposed for so long that we observe several positions for moving objects. Sometimes this is on purpose for artistic reasons (left), while other times it is unintentional (right).

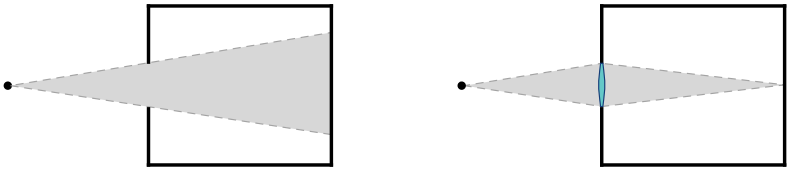


Figure 2.6: When introducing a lens to the aperture, we can focus the light rays passing through the aperture onto a single point on the imaging plane, which means that we have more light without the loss of sharpness.

By widening the aperture, we cause more light to reach the image plane, but this causes another problem: the resulting image is blurred as rays emanating from a single point reach a large region of the image plane rather than a single point; see figure 2.4. We would therefore need to choose between a brighter image with blur or a darker but sharper image. One option is to simply image the scene for a longer time, so that more light photons strike the sensor. However, that is obviously not possible for moving scenes. Therefore, we instead introduce lenses to the camera system.

2.1.2 Camera lenses

For moving scenes, we can only briefly expose the captured scene; otherwise light rays emanating from different positions of the same object, or even from different objects, will reach the same location on the image plane. This results in motion blur as in figure 2.5. As discussed earlier, we would not need to image the scene for a long time if we were able to use larger apertures that let through more light; but such cause blur themselves. By introducing lenses into the imaging system we are able to focus all the light emitted by a single point in space, through a larger aperture, onto a single point on the imaging plane, as shown in figure 2.6.

A lens aims to focus all of the rays from a given scene point to a single point

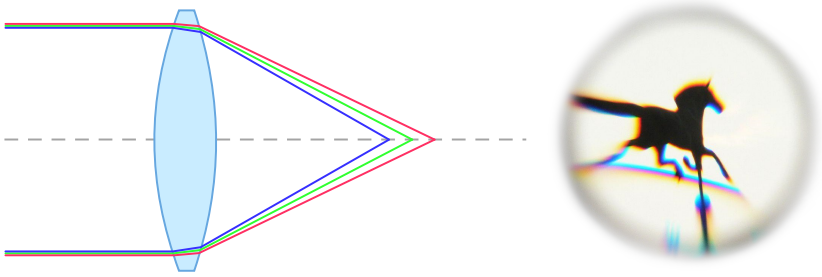


Figure 2.7: Chromatic aberration: an illustration of the effect of Snell's law (left) and an extreme example of chromatic aberration (right).

on the image plane. However, lenses have limitations and can only correctly focus light bundles from points at a predetermined distance: the focal length of the lens. Rays emitted by points at a different distance will get focused to a point either in front or behind the image plane, causing blurry images after all: so-called focal blur. However, the amount of focal blur is far less than that which would have been caused by the larger aperture. In practical cameras, the focal length can be adjusted by moving the lens forward and backwards. Construction defects in the lenses or the alignment between the sensor plane and the lens can cause a variety of additional distortions in the resulting images (see section 2.1.3.3), which often manifest as additional blur. Realistic solutions are built with multiple lenses, so that one lens compensates for most of the distortions of the others.

In practice, a pinhole camera is only used when lenses are not possible, e.g., for gamma-ray cameras. They are not applicable for moving objects, where the imaging time would be too long and significant motion blur would result; in the context of computer vision and more specifically robotics and automation, pinhole cameras are simply too impractical and cameras with lenses are used nearly without exception. However, there is one additional drawback to the use of lenses. According to Snell's law, the refraction of a light ray passing through the lens is dependent on its wavelength. This means that light rays of different colors will be offset slightly after passing through the lens: this offset will be visible as so-called chromatic aberration in the images [Mallon 2007], as shown in figure 2.7. Contrary to the earlier discussion of lens distortions, this is not a result of manufacturing errors, but simply a result of the laws of physics. However, this effect, too, can be mitigated by using camera lens systems comprised of several separate lenses, as discussed in section 2.1.3.3.

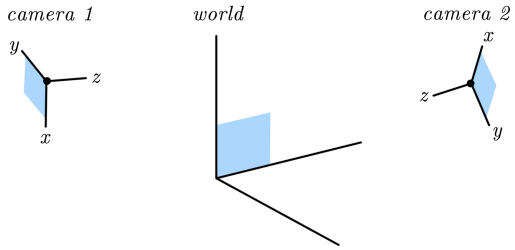


Figure 2.8: Illustration of the reference systems in a multi-camera scenario: the common, world, reference system is used to express the final reconstruction in. Each of the cameras first transform point coordinates to their own reference system before applying their projection operations.

2.1.3 Perspective cameras

In order to express the properties of lenses parametrically, and to facilitate the estimation of these camera parameters, we express the relationship between a homogeneous 3D point $\boldsymbol{p} = [x, y, z, 1]^{\mathrm{T}}$ and its projected homogeneous 2D point $\boldsymbol{u} = [u, v, 1]^{\mathrm{T}}$ mathematically. We identify four distinct steps:

1. the transformation from the point's 3D coordinates in the world coordinate system to its 3D coordinates in the camera's reference system (using the extrinsic camera matrix),
2. the perspective projection onto the image plane, resulting in 2D image coordinates,
3. the distortions introduced by imperfect lenses (using the lens distortion coefficients), and
4. the mapping of the coordinates on the image plane's reference system to those on the sensor's reference system (using the intrinsic camera matrix).

We discuss each of these steps in turn, to arrive at the fully parametrized perspective camera model.

2.1.3.1 The extrinsic matrix

In 3D, we have two reference systems: the world reference system, in which point locations are expressed, and the camera reference system, which is used for the subsequent perspective projection below. These coordinate systems are related by a rigid transformation, i.e., a rotation and a translation, which define the camera's position and orientation in the world, as illustrated in figure 2.8. Given the rotation matrix R and the translation vector $\boldsymbol{t}$ that relate both coordinate systems, we can express the point's coordinates in the camera coordinate system

$\boldsymbol{p}_c = [x_c, y_c, z_c, 1]^{\mathrm{T}}$ in terms of its world coordinates $\boldsymbol{p}$ and the so-called extrinsic matrix $[R \mid \boldsymbol{t}]$ [Zhang 2000]:

$$\boldsymbol{p}_c = [R \mid \boldsymbol{t}]\, \boldsymbol{p}. \tag{2.1}$$

From this extrinsic matrix we can also easily extract the camera location $\boldsymbol{c}$ in world coordinates: it is the point that gets mapped onto the origin of the camera coordinate system:

$$\begin{aligned} \begin{bmatrix} \mathbf{0} \\ 1 \end{bmatrix} &= [R \mid \boldsymbol{t}]\, \boldsymbol{c} \\ \implies \boldsymbol{c} &= \begin{bmatrix} -R^{\mathrm{T}}\boldsymbol{t} \\ 1 \end{bmatrix} \end{aligned} \tag{2.2}$$

2.1.3.2 Perspective projection

Having expressed the point $\boldsymbol{p}$ in the camera coordinate system as $\boldsymbol{p}_c$, we now project it onto the image plane [Zhang 1999], given by $z = 1$, by simply dividing by its z-coordinate z_c, yielding the undistorted image plane coordinates $\boldsymbol{u}_n = [u_n, v_n, 1]^{\mathrm{T}}$:

$$\boldsymbol{u}_n = \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix} = \begin{bmatrix} x_c/z_c \\ y_c/z_c \\ 1 \end{bmatrix} \propto \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{p}_c. \tag{2.3}$$

2.1.3.3 Lens distortions

Up to this point, we have assumed an ideal pinhole camera model, i.e. a camera with perfect lenses that are able to focus all points well. However, various imperfections, e.g., in the construction of the lenses themselves or in the assembly of the camera, result in distortions of the images.

One source of errors is a misalignment between the lens plane and the image plane. If they are not parallel, as in figure 2.9, tangential distortion or decentering distortion occurs [Brown 1966, Conrady 1919]. An extreme example of the resulting distortion is shown in figure 2.10. In modern cameras, the construction precision is high, such extreme distortions are very rare.

Another source of problems are imperfections in the lens itself. These distortions are modeled by the five Seidel aberrations as illustrated in figure 2.11: spherical aberration, coma, astigmatism, curvature of field and radial distortion.

In order to mitigate these Seidel aberrations, complex lens systems are required; they can contain up to dozens of different lenses (see figure 2.12). Such solutions are typically reserved for high-end consumer or professional-grade cameras. It is manageable to eliminate the first four Seidel aberrations, but radial distortions often remain an issue. We therefore assume that only radial and tangential distortions need to be modeled, as is typical in the literature [Zhang 2000].

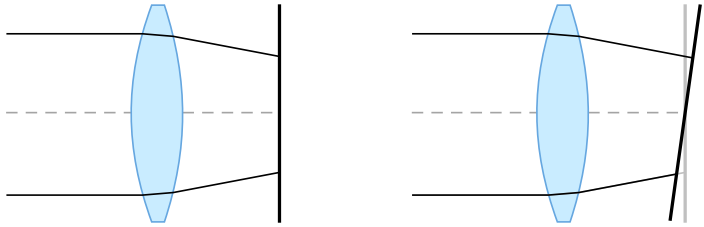


Figure 2.9: Tangential distortion is caused by a misalignment between the lens and the image plane. Left: no tangential distortion, as the image plane is parallel with the lens plane. Right: tangential distortion, which causes projections farther from the optical center to shift towards the bottom more.

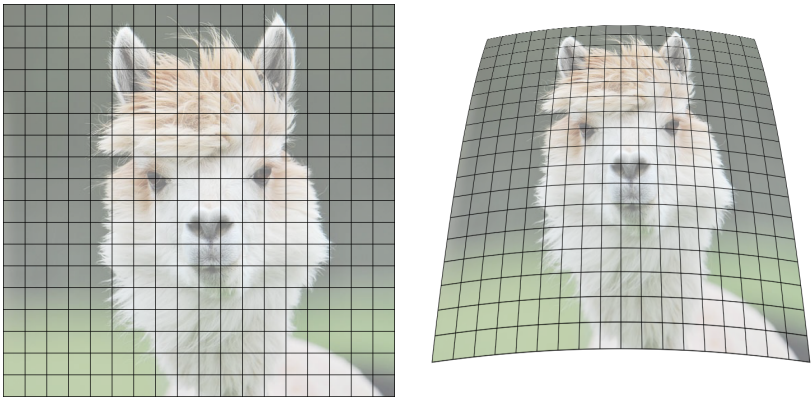


Figure 2.10: An extreme example of tangential distortion, to illustrate the effect. On the left the original, undistorted image, and on the right the noticeably distorted version. Note that the degree of misalignment was greatly exaggerated in the figure, for visual effect; most consumer cameras have non-zero but relatively low amounts of this type of distortion.

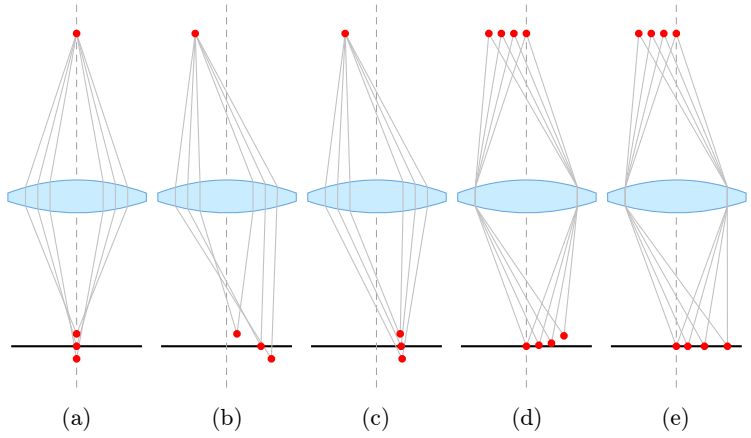


Figure 2.11: Illustration of the five Seidel aberrations. Scene points (top) are projected onto the image plane (bottom). From left to right: (a) spherical aberration: rays are never perfectly focused, (b) coma: rays from points not on the main axis are not focused well, (c) astigmatism: rays from points not on the main axis are only focused in one direction, (d) curvature of field: rays not from points on the main axis are focused, but not on the image plane, and (e) radial distortion: all rays are focused on the image plane, but it may not be on the right spot.

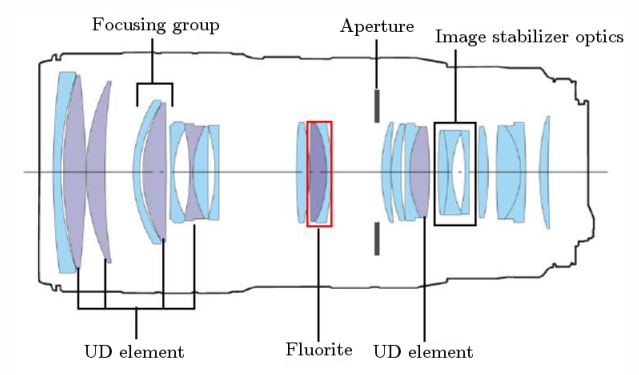
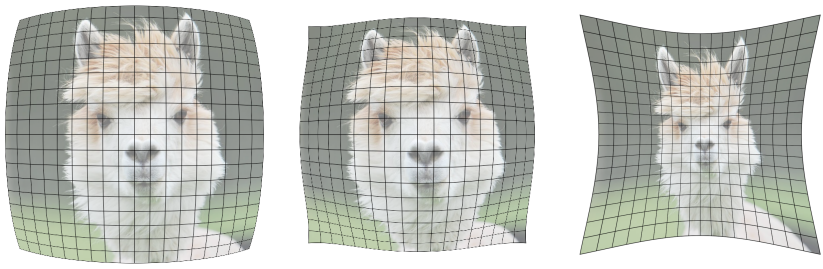


Figure 2.12: A diagram of a Canon EF 70-200mm (f/2.8) lens. Note that there are lenses both before and after the actual aperture.



Barrel distortion Moustache distortion Pincushion distortion

Figure 2.13: The various radial distortions that can occur. While the barrel and pincushion distortion are dominated by quadratic terms, the moustache distortion suffers from significant fourth-order terms - at the center, it resembles barrel distortion while at the edges it is more like pincushion distortion. The name is derived from the upper lines looking like a handlebar moustache.

Radial distortions are subdivided in three classes: barrel distortion, pincushion distortion and moustache distortion, depending on which effect the distortion has on the image, as shown in figure 2.13. For barrel distortion, the image is bloated outwards, whereas pincushion sees it pinched inwards. In case of moustache distortion, the center is bloated while the edges are pinched; the reverse is technically also possible, but does not occur in practice.

We now model these aberration mathematically, as a transform of the undistorted image plane coordinates $\boldsymbol{u}_n$ to their distorted version $\boldsymbol{u}_d$.

$$\boldsymbol{u}_d = \boldsymbol{f}_t(\boldsymbol{f}_r(\boldsymbol{u}_n)), \tag{2.4}$$

where $\boldsymbol{f}_t(\cdot)$ and $\boldsymbol{f}_r(\cdot)$ define the tangential and radial distortion, respectively, which are applied one after the other. In the universally adopted Brown-Conrady plumb-bob formulation [Conrady 1919, Brown 1966], they are given by:

$$\begin{aligned} \boldsymbol{f}_t([u,v]^{\mathrm{T}}) &= \begin{bmatrix} u + [2k_1uv + k_2(r^2+2u^2)] \\ v + [k_1(r^2+2v^2) + 2k_2uv] \end{bmatrix}, \\ \boldsymbol{f}_r([u,v]^{\mathrm{T}}) &= \begin{bmatrix} u \\ v \end{bmatrix} (1 + k_3r^2 + k_4r^4 + k_5r^6), \end{aligned} \tag{2.5}$$

where we have introduced $r = \sqrt{u^2+v^2}$ for notational brevity. This distortion model depends on only five coefficients: $\boldsymbol{k} = [k_1, k_2, k_3, k_4, k_5]^{\mathrm{T}}$. Because r is typically small, roughly in $[0; 0.5]$, the lower order terms dominate these distortions and it is often assumed that $k_5 = 0$ to simplify the model [De Villiers 2008].

2.1.3.4 The intrinsic matrix

The final transform is from the coordinates $\boldsymbol{u}_d$ on the image plane to the coordinates $\boldsymbol{u}_p$ in the sensor reference system – the pixel coordinates as in section 2.2.

There are three parts to this mapping: a scaling due to the focal depth, an offset because of a differing origin of both coordinate systems, and possible a skew because the sensor reference system is not necessarily an orthogonal one. We discuss each of these terms in turn, and finally construct the intrinsic matrix K that collects their effects.

It is not necessary to model a rotation in this mapping; we will assume that the x-axis of the sensor coordinate system is parallel to that of the image plane; any rotation has been modeled by the extrinsic matrix $[R \mid \boldsymbol{t}]$. Furthermore, the image distortion is invariant to a rotation of the image plane coordinate system, as it depends only on the distance to the origin $r = \sqrt{u_n^2 + v_n^2}$.

By placing the image plane at $z = 1$ in the perspective projection step, we have implicitly assumed the focal depth to be 1. In order to allow for different focal lengths, we scale the image coordinates $\boldsymbol{u}_d$ with the matrix

$$\begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

where f is the focal length expressed in terms of the size of a pixel.

For practical and historical reasons, digital images are indexed using a coordinate system that has its origin in the top left corner, as in figure 2.14. Because the origin of the normalized image coordinates is somewhere in the center of the image, we require a translation $[u_0, v_0]^{\mathrm{T}}$ in the image domain:

$$\begin{bmatrix} 1 & 0 & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.7}$$

Finally, if for some reason the sensor coordinate axes are not perpendicular, there may be significant skew as shown in figure 2.15 for a hexagonal sensor grid [Staunton 1989, Bell 1989]. This is modeled by a skew matrix:

$$\begin{bmatrix} 1 & \alpha & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.8}$$

where α denotes the extent of the shear. The skew factor is rarely significant, and in practice it is often set to zero.

The impacts of the focal length, the optical center translation and the camera skew are combined into the intrinsic matrix K [Zhang 1999], such that the digital image coordinates $\boldsymbol{u}_p$ are given by

$$\boldsymbol{u}_p = K\boldsymbol{u}_d = \begin{bmatrix} 1 & 0 & u_0 \\ 0 & 1 & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & \alpha & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_u & 0 & 0 \\ 0 & f_v & 0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{u}_d. \tag{2.9}$$

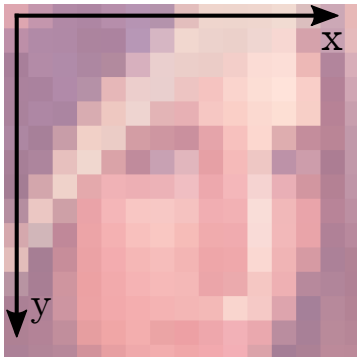


Figure 2.14: For practical and historical reasons, digital images are indexed using a left-handed coordinate system with its origin at the top left corner.

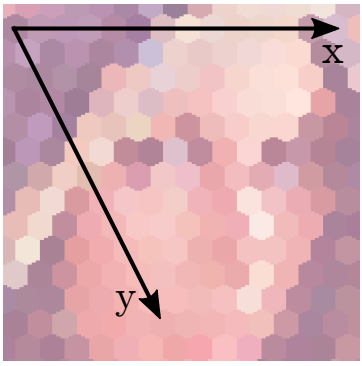


Figure 2.15: In some rare cases, the axes of the image are not modelled as parallel, which can be easier, e.g., for hexagonal pixel grids.

While the extrinsic matrix changes drastically any time the camera moves through space, the lens distortions and the intrinsic matrix are generally fixed. Only the focal length f tends to change: many consumer models have variable focus, which is controlled by an auto-focus module or manually by the user to focus the camera at various distances. In practice we often estimate the intrinsic parameters and lens distortions in advance, so that we only need to calibrate the cameras' extrinsic parameters and possibly their focal depth in the field.

2.1.4 Orthographic cameras

Perspective cameras image a scene by capturing only those light rays that pass through the aperture. On the other hand, orthographic cameras, also known as telecentric cameras, only capture those rays that are parallel to the viewing direction. Due to the practical problems of achieving this, they are much more difficult and expensive to construct, mainly stemming from the fact that the image plane (i.e. the collection of sensors) of orthographic cameras must be at least as large as the largest object being imaged, or that the sensor array must be moved over the entire scene without that scene changing. In practice, they are only used in cases where cameras are not possible, such as in the case of the hyperspectral linear pushbroom cameras discussed later.

A practical example is a flat-bed scanner, which moves a linear sensor array over the entirety of a page. For computer vision, however, orthographic cameras are rarely practical. We include a discussion on the orthographic model for two reasons: it is used as an approximation of the perspective model in chapter 7 as it has the advantage of being completely linear, and it is also a constituent of the

next camera model we discuss, the linear pushbroom camera.

For orthographic cameras, the points are transformed into the camera coordinate system as in the perspective model, but then the projection onto the image plane is a parallel one, i.e. we simply discard the Z coordinate and use the X and Y coordinate as the projection. For this camera model, lens deformations are typically ignored. Given the point $\boldsymbol{p}$ in the world coordinate system, we calculate its coordinates $\boldsymbol{p}_c$ in the camera coordinate system using the same extrinsic matrix $[R \mid \boldsymbol{t}]$ as in the perspective case. The subsequent projection $\boldsymbol{u}_n$ on the image plane is then performed by a matrix that simply discards the Z coordinate:

$$\boldsymbol{u}_n = \begin{bmatrix} u_n \\ v_n \\ 1 \end{bmatrix} = \begin{bmatrix} x_c \\ y_c \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [R \mid \boldsymbol{t}]\, \boldsymbol{p}. \tag{2.10}$$

After the projection on the image plane, we use an intrinsic matrix similar to the one in the perspective case to transform the image plane coordinates $\boldsymbol{u}_n$ to pixel coordinates $\boldsymbol{u}_p$ in the sensor's reference system. Similar to the perspective camera model, we have an intrinsic matrix K which contains scaling factors for the X and Y coordinates (expressing the focal distance in terms of the pixel size), and the optical center of the image plane $[u_0, v_0, 1]^{\mathrm{T}}$:

$$K = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

The whole projection model is then given by the following linear relationship [Li 2013] (as we no longer have the proportionality and do not need to divide by the z-coordinate for normalization of the homogeneous point):

$$\boldsymbol{u}_n = \begin{bmatrix} f & 0 & u_0 \\ 0 & f & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} [R \mid \boldsymbol{t}]\, \boldsymbol{p}. \tag{2.12}$$

We end this section with a note on how an orthographic camera can approximate a perspective one. For the projection of a set of points, the difference between the perspective camera and the orthographic camera is that the former divides the points by their Z-coordinate, while the latter simply discards this coordinate. In the theoretical case that the Z-coordinates of all the projected points are the same, $\bar{z}$, then this projection could be replicated exactly by an orthographic camera where we set the focal distance to be $f/\bar{z}$. Then, if we know that all points are at *roughly* the same distance from the camera, then we can approximate the perspective projection well by scaling by the average distance over the point set. This means that, for point clouds sufficiently far from the camera compared to the range of z-values of their points, an orthographic camera model is a valid approximation, as illustrated in figure 2.16. If this approximation is valid – and we argue in chapter 5 and

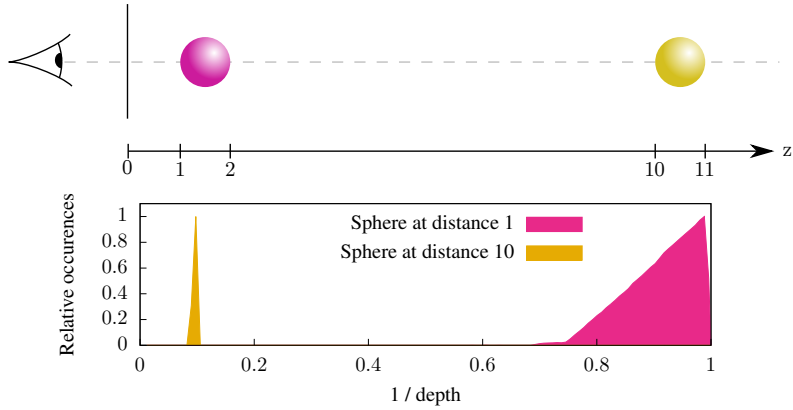


Figure 2.16: The inverse depth distribution of two objects in the scene, one close to and one far away from the camera. The difference between an orthographic camera and a perspective one is the depth-dependent scaling of the image points. When the inverse depths, i.e. the scaling factors, are nearly constant and hence well approximated by a single value $\bar{z}$, then the orthographic camera model is a good approximation for a perspective camera.

chapter 7 that, in some situations, it is – the projection can now be represented by a simple linear relationship.

2.1.5 Linear pushbroom cameras

Linear pushbroom cameras are a mixture of orthographic and perspective cameras: orthographic in one image coordinate, perspective in the other [Gupta 1997, Draréni 2011]. This is the result of passing a linear sensor array over a scene as in figure 2.17: in the direction of the array movement the projection is orthogonal, while it is a perspective projection in the other direction. For simplicity's sake, we will assume that the movement is orthogonal to the sensor array, as in figure 2.17. In this sense, one of the directions (the perspective one) is spatial, while the other (the orthographic one) is temporal. Because of this, this type of cameras is only really useful for static scenes, and the temporal dimension makes no difference: we interpret it simply as a spatial one. There are two main scenarios where this type of camera is used: satellite imaging and hyperspectral cameras (see section 2.3).

Linear pushbroom cameras are common in satellites for two reasons: weight and orbital physics. On the one hand reducing volume and weight is very important for space applications; as the linear pushbroom cameras consist of only a linear sensor array rather than a full 2D sensor grid, they can be smaller and lighter than traditional cameras. Furthermore, the possible lens distortions over the 1D array are much less complex, so that the lenses can also be simpler and more lightweight. On the other hand, their movement can, at least locally, be assumed to be parallel

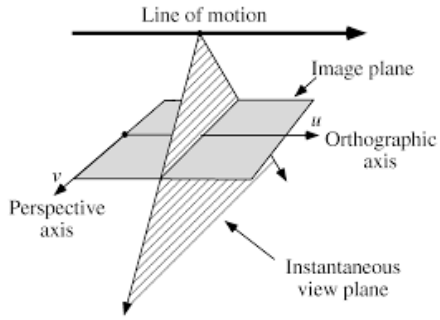


Figure 2.17: Illustration of a linear pushbroom camera. A linear sensor is moved over a scene, capturing a single image line at any given instant. In that sense, one of the directions (the perspective one) is spatial, while the other (the orthographic one) is in practice temporal – but as we image static scenes, the distinction is not important.

to the ground and so orthogonal to the looking direction: they are naturally a good fit for the linear pushbroom model.

For hyperspectral cameras, a linear pushbroom camera is required for practical reasons as discussed in section 2.3.2: such hyperspectral cameras use a prism or grating to disperse the wavelengths of a single-line measurement over a traditional 2D sensor grid, and then move this measurement line over the scene to capture the hyperspectral volume.

The projection performed by linear pushbroom cameras was first modelled mathematically in literature by [Gupta 1997]. Their model relates the 3D point $\boldsymbol{p}$ to the 2D location $\boldsymbol{u}_p$ on which it is projected. Lens deformations are ignored: as the lenses in these systems only need to correct distortions along one dimension rather than two, they are much easier to produce and do not result in strong distortions. Similar to the orthogonal case, we then have three steps left: transforming the world coordinates to the camera coordinate system, projecting the points, and applying an intrinsic matrix to map the projected points onto the sensor coordinate system.

We express the camera's movement direction and speed in its coordinate system as $\boldsymbol{v} = [v_x, v_y, v_z]^{\mathrm{T}}$. A point $\boldsymbol{p} = [x_c, y_c, z_c]^{\mathrm{T}}$ in the camera coordinate system is therefore under the sensor line (which is at $x_c = 0$) imaged at time x_c/v_x, at which point it is at location $\boldsymbol{p}_i = [0, y_c - x_c v_y/v_x, z_c - x_c v_z/v_x]^{\mathrm{T}}$ in the camera coordinate system. Similar to earlier models, we first transform the point's world coordinates $\boldsymbol{p}$ to the camera coordinate system with the extrinsic matrix $[R \mid \boldsymbol{t}]$, so that the full transform becomes [Gupta 1997]:

$$\boldsymbol{p}_i = \begin{bmatrix} 0 & 0 & 0 \\ -v_y/v_x & 1 & 0 \\ -v_z/v_x & 0 & 1 \end{bmatrix} \boldsymbol{p}_c = \begin{bmatrix} 0 & 0 & 0 \\ -v_y/v_x & 1 & 0 \\ -v_z/v_x & 0 & 1 \end{bmatrix} [R \mid \boldsymbol{t}]\, \boldsymbol{p}. \tag{2.13}$$

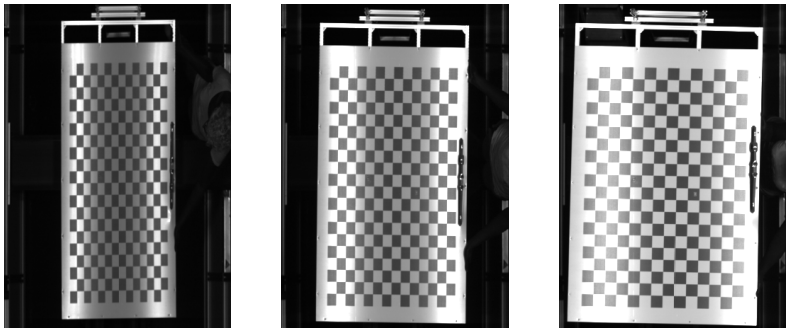


Figure 2.18: Example of images captured with a linear pushbroom: the same checkerboard is imaged at different distances from the camera, but only the projections on the horizontal axis is affected by the change in distance.

As mentioned before, the projection of $\boldsymbol{u}_n = [u_n, v_n]^{\mathrm{T}}$ is a mixture of orthogonal, for the first coordinate u_n, and perspective, for the second coordinate v_n:

$$\begin{aligned} u_n &= x_c / v_x, \\ v_n &= y_i / z_i = \frac{y_c - x_c v_y / v_x}{z_c - x_c v_z / v_x}. \end{aligned} \tag{2.14}$$

As a final step we have, as in previous models, an intrinsic matrix K to map from the image plane coordinates $\boldsymbol{u}_n$ to the location $\boldsymbol{u}_p$ in the sensor reference system:

$$\boldsymbol{u}_p = K \boldsymbol{u}_n = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{u}_n, \tag{2.15}$$

which concludes the mathematical formulation of the linear pushbroom model.

We note that an object's distance to the camera only affects one coordinate of the projection as seen in figure 2.18: objects that get closer to the camera will only see their projection grow along the Y-axis (although the images were transposed for logistic reasons).

2.1.6 Plenoptic cameras

Finally, we discuss plenoptic cameras, which will be used in chapter 8. In essence, they are a large collection of very small perspective cameras as seen in figure 2.19. The image plane is now the combination of the image planes of all these small cameras organized in a (usually rectangular) grid, as seen in figure 2.20. In practice, such cameras still use a main lens to focus all the light from the scene, in addition to the plane of microlenses, which represent the virtual "pixels". Each of these microlenses has its own small image plane, and it focuses the incident light

Figure 2.19: An image taken by a plenoptic camera, where the separate images of each of the microlenses are clearly visible.

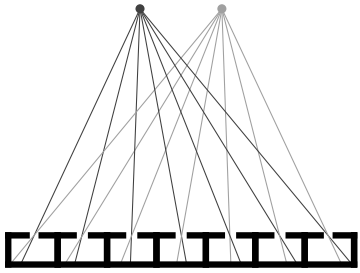


Figure 2.20: Plenoptic cameras are built up from many small perspective cameras, each of which focuses light on its own part of the image plane without interference by the others.

on that. An alternative interpretation is that a plenoptic camera not only measures light intensity at its "pixels" (the microlenses), but also distinguishes between light coming from various directions: in a traditional camera the measurements would be the sum of all incident light on each microlens.

There are two variants of such plenoptic cameras: unfocused and focused plenoptic cameras. The first type focuses the light onto the plane of microlenses [Ng 2005]. The second one is not restricted to this: it could focus the incident light on an image plane in front or behind the plane of microlenses [Lumsdaine 2009, Perwass 2012]. In the former, the assumption is that each microlens image is completely defocused with respect to the image created by the main lens (as they are focused on their own plane, where the main lens focuses its image), and only a single pixel in the final image can be rendered from it, which results in relatively low resolutions. In focused plenoptic cameras, the microlens array is interpreted as an imaging system focused at the image plane of the main lens, which no longer coincides with the microlens plane. By varying the focus of the main lens (and adapting that of the microlenses to match), they allow a trade-off between the spatial resolution of the resulting images and the angular resolution of the lightfield.

2.2 Analog pictures and digital images

Initially, cameras captured images on analog photosensitive film. However, in order to store the information digitally, and to be able to process them efficiently, digital cameras see the photosensitive film on the image plane replaced by a discrete grid of photosensitive components. These components produce an electric signal proportional to the intensity of light that strikes them; each component represents a single picture element or *pixel*, as in figure 2.21, and the size of the pixel

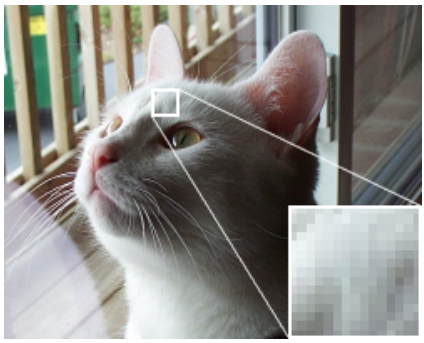


Figure 2.21: Digital images are built from pixels; zooming in on the images will make the pixel boundaries clearly visible.

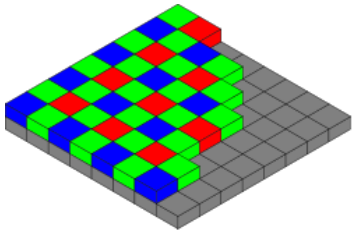


Figure 2.22: The Bayer filter array. This is the most common colour filter array deposited on a sensor array. The filter has twice as many green filters as red or blue ones in order to mimic the human physiology, which has a higher effective spatial resolutions for green.

grid is directly related to the image resolution.

Each pixel, measuring the intensity of the light striking its surface area, represents a single sampling location on the image plane. Here, we face a trade-off related to light quantity and noise: while larger pixels mean that more light falls on the sensors, it also means that fewer pixels fit in the same camera. Larger pixels also mean that we are not able to capture fine details of the scene as the spatial resolution of the sampling grid is decreased. When using smaller pixels we can amplify the measured light intensity more, but due to the discrete nature of light's photons and the corresponding random process this also amplifies the image noise.

The light sensors make no distinction on the color of the incident light: all light is counted equally. Therefore we require color filters on top of the sensors in order to capture color information about the scene. As nearly all sensors are opaque, we cannot place multiple sensors on top of one another, and so we can only have each pixel measure a single color. For this reason, a grid of color filters is overlaid on top of the sensor grid, as in the Bayer filter of figure 2.22 [Bayer 1976]. The missing colors are filled in by dedicated demosaicing algorithms that combine the information from all sensors to provide a full-resolution full-color image [Kimmel 1999, Li 2008]. Various sampling grids exist, but the Bayer grid is the most used: it mimics the physiology of our eyes, which are more sensitive to high spatial resolution in green than either red or blue. At least one producer (Foveon) produces translucent sensors that allow a pixel to estimate all three channels simultaneously. However, these sensors are relatively large, and thus trade off spatial resolution for color resolution. Another alternative is to split the different color channels with the use of prisms, which is what happens in 3CCD cameras, which yields a full-resolution measurement of all color channels, at the cost of three times the photosensors as well as the requirement for an accurate prism set-up. Testa-

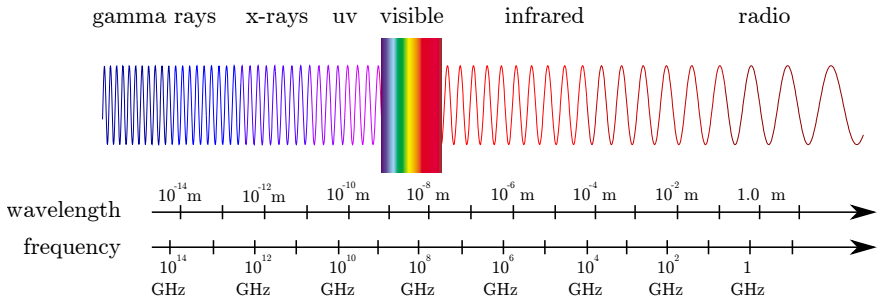


Figure 2.23: The entire electromagnetic spectrum. Note how the human eye only sees a small fraction of the spectrum: the visible spectrum.

ment the fact that many image compression algorithms intentionally downsample colors (because the human vision system is much more sensitive to intensity changes than color changes), the trade off is often not found worth it, and consumer cameras with these translucent sensors or three separate photosensor grids are rare.

2.3 Imaging modalities

Depending on the scene we observe and the application we have in mind, we wish to capture different types of information about the scene. Most of the time, we will capture color images in the traditional three channels: red, green and blue. Such images reflect the reality as we observe it: they can accurately represent the light waves in that part of the spectrum that is visible to the human eye. Yet as figure 2.23 shows, the visible spectrum is only a small part of the electromagnetic spectrum; while we may not be able to see most types of waves, they often offer invaluable information about the world around us. Here we briefly introduce those modalities (the collection of measured wavelengths) that will be returning later in this thesis: both shortwave and thermal infrared, as well as hyperspectral cameras. We also include a discussion on time-of-flight cameras. Although such cameras don't actually measure light emitted by the scene (but rather the reflections of pulses emitted by the camera itself), it is often interpreted as "just another imaging modality".

2.3.1 Infrared cameras

Infrared cameras capture wavelengths that are longer than can be seen by humans: either just outside of the human range (shortwave infrared) or way past it (thermal infrared).

Shortwave infrared (SWIR) is not visible to our eyes, but it behaves similar to visible light. That is, SWIR light is reflective: it bounces off of objects much like visible light does, and the resulting intensity images are similar to those captured by a traditional camera. However, it is possible to synthetically brighten the images by introducing an infrared light source. Most terrestrial organisms don't react much to this type of light, as we have had little evolutionary reason to do so: such an artificial light source influences the scene much less than normal light bulbs would. For example, think of security cameras: they can use infrared LEDs to brighten up footage without annoying every passer-by or alerting would-be trespassers.

On the longer wavelength side of the infrared spectrum, thermal infrared waves are mostly emitted by an object because of its temperature; a specially designed camera can measure those waves striking its sensor plane in order to perform non-invasive temperature measurements. Two important facts combine to make thermal infrared a very interesting modality: each object emits its own thermal radiation, and objects barely reflect incident thermal radiation (instead, they absorb it and heat up slightly because of it). This means that we don't need an external wave emitter in order to image a scene, in contrast to traditional photography where only a small number of objects emit light themselves (such as the sun and other stars, or candles) light has to be provided by a ternary source in order to image other objects.

2.3.2 Hyperspectral cameras

So far, the discussions of imaging modalities have assumed that a very small number of wavelengths are measured. For hyperspectral cameras, the goal is to capture a large number of wavelengths for each pixel in the image. In practice, a pixel sensor can only measure a single intensity value. Often this is the combination of a predetermined set of wavelengths as enforced by a wavelength filter. For a low number of wavelengths, we can overcome this limitation by using an approach like the Bayer grid and a demosaicking algorithm. However, an approach like the Bayer grid becomes impractical for hundreds of wavelengths as the resulting spatial resolution would be too coarse.

For that reason, hyperspectral imaging systems work differently [Chang 2003, Fischer 2006]. The output of a hyperspectral camera is three-dimensional: we measure an intensity for each wavelength λ at each pixel position (u, v). There are three main ways to do this:

1. Capturing single-wavelength images of the scene and changing the captured wavelength over time (spectral scanning) [Lu 2014],

2. a generalization of the Bayer filter with the discussed drawbacks (snapshot scanning) [Hagen 2012], or

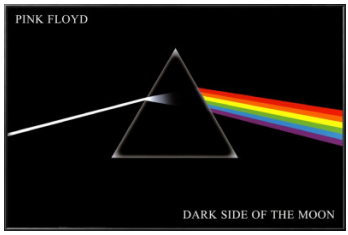


Figure 2.24: Pink Floyd's Dark Side of the Moon album cover, illustrating diffraction.

3. using a pushbroom camera to diffract a single line of the scene into its different wavelengths (spatial scanning) [Lu 2014].

We are most interested in the latter of the three: while the generalized Bayer filter approach is much faster than the subsequent capture of each wavelength in a spectral scanning camera, it results in very low resolution images. On the other hand, a spectral scanning camera is prohibitively expensive and difficult to manufacture. The pushbroom approach strikes a good middle ground and has gained much popularity.

As famously illustrated by the Pink Floyd album cover (figure 2.24), different wavelengths exit a prism under different angles due to Snell's law mentioned earlier in relationship with chromatic aberration in lenses. Pushbroom hyperspectral cameras use this to split the incoming light into its constituent wavelengths. A single line of the scene is captured at a given time, and the wavelengths are spread out orthogonally to this line. This means that a single 2D sensor array is sufficient to measure all wavelength information for this single line, and by moving this line over the scene we can acquire information for all of it.

Furthermore, properties of prisms and gratings for splitting light bundles are well known. This means that with proper design the sensor array does not need any special or expensive wavelength filters, contrary to spectral scanning. Moving the camera over the scene is often much easier than dynamically changing the wavelength filters on the sensor array at a high rate, which is why pushbroom cameras are often the preferred technique for hyperspectral captures. The only drawback is that the camera has to be able to move over the entire scene.

2.3.3 Time-of-flight cameras

Time-of-flight (ToF) cameras are cameras that don't capture specific wavelengths, but instead actively measure the distance to each sample point in the scene using infrared rangefinders [Gokturk 2004]. By emitting periodic shortwave infrared pulses, they actively and directly influence the scene they are measuring; these emitted pulses reflect on the objects in the scene and from the measured round trip

Figure 2.25: The effect of Gaussian noise on both natural and artificial images. Gaussian noise adds as much high-frequency noise as it does low-frequency perturbations (theoretically, its spectral power function is a constant): if there were no high-frequency components to start with, as in the synthetic image on the left, those high-frequency artifacts will be much more noticeable.

time (RTT) of the pulses, the distance of the object to the camera. Depending on the interpretation, they might not be cameras in the traditional sense of the word, as they do not measure emitted light, but in chapter 5 we model them as such. This allows us to leverage the mathematical projection models from chapter 2 to relate locations in the world coordinate system to measurements on their pixel grid.

While such cameras are useful because they offer a direct measurement of depth from the scene, they have three important drawbacks:

1. They are limited by the intensity of the light they emit. If the pulses are not bright enough compared to the ambient light, it may not be possible to accurately detect the reflected pulse.
2. As each pixel needs its own light pulse, and therefore its own laser, the pixels become very large. As a result, the resolution of the images is typically relatively low.
3. As light travels at, well, light speed, the resulting RTT is very low and can be challenging to accurately measure. Time-of-flight depth images are often also rather noisy.

2.4 Image noise

We consider three sources of image noise: the discrete nature of light, thermal and electrical noise, and quantization noise from the analog-to-digital conversion.

Due to the quantized nature of photons, measuring light intensity boils down to counting separate photons, which is a stochastic process that can be modeled with a Poisson distribution. To accurately estimate the intensity of light striking a sensor when that light is faint, we would have to measure the incident light over

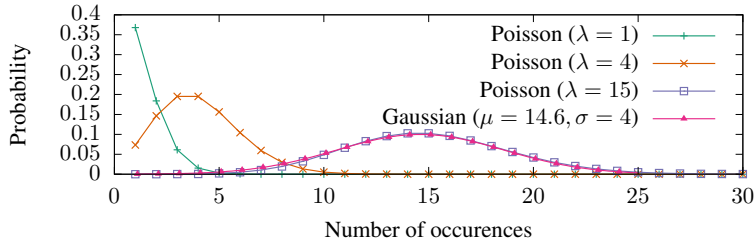


Figure 2.26: Poisson and Gaussian distributions. For large expected values (i.e. high intensity), Poisson noise resembles Gaussian noise. As the sum of two Gaussian processes is again Gaussian, we typically model the image noise as only one single Gaussian process.

either a very long time (lowering the temporal sampling frequency) or a very large area (lowering the spatial sampling frequency): neither is an interesting option. As a result, images captured in low-light settings often contain Poisson noise. Additionally, there is also Gaussian noise from internal thermal and electrical noise inside the sensors, whose impact is shown in figure 2.25. Finally, the resulting values get quantized to represent them in a finite number of bits, which adds (typically uniform) quantization noise.

In order to simplify the noise model, we take into account that a Poisson process with sufficient observation time or sufficient intensity is well modeled by a Gaussian process, see figure 2.26. At the same time, we will assume that the quantization noise is low enough to be ignored, and therefore model the image formation process as Gaussian whenever we require a probabilistic model.

It is important to make the distinction between the image noise discussed here (measurement noise on the image intensities), and the measurement errors of observed locations, which are due to the detector inaccuracies. For a discussion of the latter, please refer to section 6.2.

2.5 Conclusions

In this chapter we have outlined the basics of the image formation process: we have discussed several camera models mathematically, as well as the most common modalities for cameras to image. We have ended with a brief discussion on the noise that often perturbs our measurements. While this chapter does not contain any novel research, it does provide an invaluable summary of and introduction to camera models and image formation, upon which the rest of this thesis builds.

3
Calibration techniques

" The fact that we live at the bottom of a deep gravity well, on the surface of a gas covered planet going around a nuclear fireball 90 million miles away and think this to be normal is obviously some indication of how skewed our perspective tends to be. "

— Douglas Adams, *The Salmon of Doubt*

In the previous chapter we have shown how the concept of camera projection can be expressed mathematically. We have derived formulas for the extrinsic transformation from the world coordinate system to a camera-specific coordinate system in which we subsequently perform projection, for the effect of lens distortion on the projected image plane points, and finally for the intrinsic transformation to express the pixel locations on the sensor grid in terms of point locations on the image plane. These transformations depend on the camera parameters: the extrinsic parameters in $[R \mid \boldsymbol{t}]$, the lens distortion coefficients in $\boldsymbol{k}$, and the intrinsic parameters in K, respectively. In this chapter we discuss the estimation of these parameters, a process called camera calibration. Based on measured 2D projections $\tilde{\boldsymbol{u}}_i$ of known 3D locations $\boldsymbol{p}_i$, we estimate the camera parameters that best explain the measurements: those values for which the simulated measurements $\boldsymbol{u}_i$ lie as close as possible to the actual measurements $\tilde{\boldsymbol{u}}_i$.

First, we discuss the ambiguity between the world coordinate system and the extrinsic camera parameters: we show that the formulas allow for a rigid transformation of the world space which lets us choose the world reference system as we wish. Then we discuss the requirements for the set of measurements, which

yields a lower bound for the number of observations required (although, in practice, more observations yield a more accurate estimate and we make as many as practical). The rest of this chapter discusses how to actually estimate the parameters: quantifying the "as close as possible" from the previous paragraph (the loss function), and finding those best parameter values (the optimization procedure). First, we discuss the calibration of perspective cameras in detail. Afterwards, we explain how to calibrate linear pushbroom cameras, with an important contribution: the preexisting work for calibrating them with checkerboards is numerically unstable for common checkerboard positions, and we have proposed an alternative calibration technique that is both stable in those degenerate positions as well as more robust in general.

3.1 The world reference system

When calibrating a camera and, more importantly, while performing the various computer vision tasks we calibrated the cameras for, we require a common reference system to express the objects and the cameras in: the world coordinate system. Typically, however, the origin or orientation of this world coordinate system is not important. Rather, we extract information such as, e.g., distances, relative movements, lengths, or areas from the world model. We now show that the calibration is only defined up to a rigid transformation: the mathematical model still allows for a rigid transform of the world without affecting the modeled measurements. We recall that, mathematically, a camera projects a 3D point $\boldsymbol{p}$ as a 2D measurement $\boldsymbol{u}$ using an intrinsic matrix K and an extrinsic matrix $[R \mid \boldsymbol{t}]$:

$$\boldsymbol{u} \propto K \left[R \mid \boldsymbol{t}\right] \boldsymbol{p}, \tag{3.1}$$

where the 3D point $\boldsymbol{p}$ is represented in the world coordinate system with homogeneous coordinates by a 4×1 vector, and correspondingly the rotation matrix R and the intrinsic matrix K are 3×3 matrices, while the translation $\boldsymbol{t}$ is a 3×1 vector. The projected 2D location $\boldsymbol{u}$ is also expressed in homogeneous coordinates, hence the proportionality sign rather than an equality sign.

Let us now apply a rigid 3D transform to the point $\boldsymbol{p}$ to yield $\boldsymbol{p}'$; we represent this transform in homogeneous coordinates by the 4×4 matrix T, which comprises a rotation matrix R_T and an offset $\boldsymbol{t}_T$: $\boldsymbol{p}' = T\boldsymbol{p}$. The projection $\boldsymbol{u}'$ of $\boldsymbol{p}'$ is equivalent to the projection of the original point $\boldsymbol{p}$ by a camera with the same

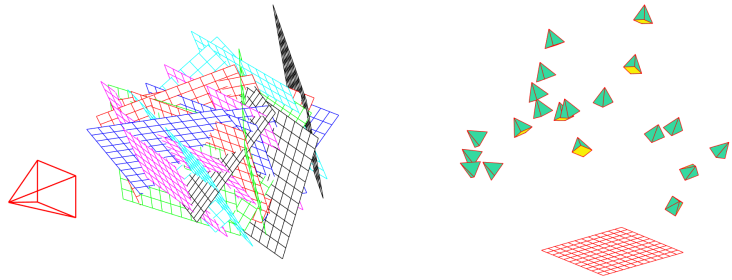


Figure 3.1: During calibration captures, the camera is typically static while the calibration object moves to different poses (left). However, this is functionally equivalent to a moving camera observing a static calibration object (right). We cannot discriminate between a movement of the camera and the inverse movement by the entire scene.

intrinsic characteristics but at a different position and with different orientation:

$$\begin{aligned}
\boldsymbol{u}' &\propto K\,[R \mid \boldsymbol{t}]\,\boldsymbol{p}', \\
\boldsymbol{u}' &\propto K\,[R \mid \boldsymbol{t}]\,T\boldsymbol{p}, \\
\boldsymbol{u}' &\propto K\,[R \mid \boldsymbol{t}] \begin{bmatrix} R_T & \boldsymbol{t}_T \\ \boldsymbol{0} & 1 \end{bmatrix} \boldsymbol{p}, \\
\boldsymbol{u}' &\propto K\,[RR_T \mid R\boldsymbol{t}_T + \boldsymbol{t}]\,\boldsymbol{p}.
\end{aligned} \tag{3.2}$$

We have omitted lens distortion from this discussion, but it does not change our conclusion: as this rigid transform is contained in the extrinsic parameters, lens distortion only comes into play after the rigid transformation has already been mitigated.

This ambiguity has a very practical application: while calibrating the camera we assume that our well known calibration object does not move, as illustrated in figure 3.1. This means that we always know the 3D location of all of its points accurately, as it is always in the same position, and instead we claim that the differing projections are caused by differing camera positions. In the left image of figure 3.1, it is impossible to accurately express the 3D locations of the various checkerboard corners, as that requires an accurate knowledge of the camera position compared to the checkerboard (which is what we are estimating in the first place). In the representation of the right figure, the 3D locations are the same in all views, and well known in advance.

After calibration, we have estimated both the intrinsic matrix and the extrinsic matrices for each of the calibration images (as well as the lens distortion coefficients), with this remaining ambiguity. To fix the world coordinate system, we fix the extrinsic matrix of one of the calibration images by calculating the transform T that aligns its estimated extrinsic matrix $[R \mid \boldsymbol{t}]$ with the chosen one $[R' \mid \boldsymbol{t}']$ and

applying this transform to all of the extrinsic matrices:

$$[R' \mid \boldsymbol{t}'] = [R \mid \boldsymbol{t}] \begin{bmatrix} R_T & \boldsymbol{t}_T \\ \mathbf{0} & 1 \end{bmatrix} \implies \begin{cases} R_T = R^{\mathrm{T}} R' \\ \boldsymbol{t}_T = R^{\mathrm{T}} (\boldsymbol{t}' - \boldsymbol{t}) \end{cases} \tag{3.3}$$

As mentioned above, this choice is irrelevant in many cases, as we are often only interested in relative distances rather than in absolute positions. In such cases, in order to get rid of ambiguity, the world reference system is often chosen to be the same as one of the camera's, i.e. $[R' \mid \boldsymbol{t}'] = [R \mid \boldsymbol{t}]\, T = [I_3 \mid \mathbf{0}]$.

3.2 Required number of observations

From the definition of the projected points, leaving out lens distortion for a second, we have the relationship

$$z_i \boldsymbol{u}_i = K\, [R \mid \boldsymbol{t}]\, \boldsymbol{p}_i, \tag{3.4}$$

which undoes the scaling of the homogeneous points $\boldsymbol{u}_i$. Horizontally concatenating the versions of this equation for all of the I measured points, we get

$$\underbrace{U P_z^T}_{3\times I} = \underbrace{K}_{3\times 3} \underbrace{[R \mid \boldsymbol{t}]}_{3\times 4} \underbrace{P}_{4\times I} \tag{3.5}$$

Recall that the intrinsic matrix is given by

$$K = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.6}$$

with $|K| = f_u f_v \neq 0$, so that K is always invertible. Therefore, we can move it to the left hand side of equation (3.5):

$$K^{-1} U P_z^T = [R \mid \boldsymbol{t}]\, P, \tag{3.7}$$

where, from equation (3.6),

$$K^{-1} = \begin{bmatrix} 1/f_u & 0 & -u_0/f_u \\ 0 & 1/f_v & -v_0/f_v \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.8}$$

We now have a system of $3I$ linear equations for the camera parameters: four parameters in the intrinsic matrix K (the focal depth in terms of the pixel width and height, the skew, and the optical center), and six extrinsic parameters (three for the orientation of the camera and three for its location). It is now important to pay attention to the possible degeneracy of the observed points $\boldsymbol{p}_i$. Due to ignorance or oversight, the observed points might all inadvertently lie in the same plane, or on

the same line. Intuitively, we suspect that this will cause trouble while estimating the camera parameters: the input does not adequately cover all possible degrees of freedom in the world coordinate system, and hence it is unrealistic to expect the calibration to infer the structure of the world coordinate system correctly.

When all points lie in a single plane, we can express all of them as a linear combination of any three non-collinear points in the set. Hence every equation in the system, except for those corresponding with the three chosen points, is redundant. Then, the linear system actually only has nine equations for ten variables and is under-determined. It is even worse if all measured points are collinear: in that case only six independent equations remain. Note that because of noise the corresponding equations will not be exactly linearly dependent in the system of equation (3.7) and it is not always straightforward to recognize this issue by the properties of this system.

We discuss the various calibration objects in chapter 4. While there are calibration objects that yield a fully 3D set (i.e. not coplanar) of point locations with a single image capture, those are hard to accurately construct. Furthermore, in practice we still need more than one image capture because of measurement noise. The ubiquitous calibration object, for its mix of accuracy and ease of use, is the checkerboard pattern any computer vision researcher is familiar with. This is a planar pattern, so we need to take care that not all of the images are of the pattern lying in the same plane (e.g. at various places on a table). Please see chapter 4 for a more complete discussion of calibration objects and their characteristics.

Finally, a note on the estimation of the lens distortion parameters. In practice, they are assumed to be zero when finding the initial estimate of the extrinsic matrices, and then optimized jointly in a refinement step. As the effect of lens distortion is more pronounced nearer to the edges of the images, it is easier to estimate the distortion coefficients from measurements near the edge of the images. For this reason, and in addition to making sure we have a measurement point cloud that covers the entire relevant 3D space, it is also interesting to measure some points that are projected near the edge of the image, insofar that such points were not yet present in the previous measurements.

3.3 Modeling the estimation problem

The mathematical foundation given hitherto builds up to a strongly motivated optimization technique for estimating the camera parameters [Zhang 2000]. We will see that due to the strong non-linearity of among others the lens distortion (see section 3.4.2), it is not possible to give a closed-form optimal solution for this problem. For this reason, an iterative optimization technique is required. All iterative optimization schemes comprise three aspects:

1. an objective function that expresses how well a given estimate explains the measurements,
2. an initial estimate, and
3. a way to calculate an update step to arrive at an improved estimate.

To quantify the accuracy of our current estimates $\hat{R}$, $\hat{\boldsymbol{t}}$, $\hat{K}$ and $\hat{\boldsymbol{k}}$, we take a look at the projections $\hat{\boldsymbol{u}}_i$ that would have resulted from these values and the known 3D locations $\boldsymbol{p}_i$:

$$\hat{\boldsymbol{u}}_i \propto \hat{K}\hat{\boldsymbol{f}}_t\left(\hat{\boldsymbol{f}}_r\left([\hat{R} \mid \hat{\boldsymbol{t}}]\boldsymbol{p}_i\right)\right). \tag{3.9}$$

If our estimates are accurate, we expect the simulated projections $\hat{\boldsymbol{u}}_i$ to be close to the measured projections $\tilde{\boldsymbol{u}}_i$.

However, in practice it is impossible to exactly match all measurements, due to measurement noise. Therefore, we express the goodness of fit $\Phi(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}})$ as the average error between the expected projections and the measured ones:

$$\begin{aligned} \Phi(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}) &= \left\| \left(\ldots, \phi_i(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}), \ldots \right) \right\|_p, \text{where} \\ \phi_i(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}) &= \left\| \hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i \right\|_q. \end{aligned} \tag{3.10}$$

Note that we have left the norms used in either the reprojection error $\phi_i(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}})$ or the aggregated error $\Phi(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}})$ unspecified.

In order to make an educated choice as to which norms to use, we consider the probabilistic formulation of the problem. Let us assume that our model of the intrinsic parameters and the lens distortion is able to completely describe the camera in question. In that case, and assuming the correct camera parameters, the only source of errors is the measurement noise in $\tilde{\boldsymbol{u}}_i$. We assume that this measurement noise is additive, independent and zero-mean; it is the realization $\boldsymbol{n}$ of a variable with probability density function $P(\boldsymbol{n})$: $\tilde{\boldsymbol{u}}_i = \boldsymbol{u}_i + \boldsymbol{n}$.

The problem we now wish to solve is the maximum likelihood estimation of the camera parameters: given the known structure of the calibration object, which values $(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}})$ for the camera parameters are most likely to have caused the observed values? Formally, this is expressed as the optimization problem

$$\max_{\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}, \hat{\boldsymbol{u}}_i} P\left(\hat{\boldsymbol{u}}_i | \tilde{\boldsymbol{u}}_i\right). \tag{3.11}$$

We now discuss three alternatives for the noise model, and the implied choice of norms in equation (3.10).

3.3.1 Gaussian noise

The noise signal is often assumed to follow a Gaussian distribution, with zero mean and independent for every observation: Additive White Gaussian Noise (AWGN).

Gaussian noise models the idea that a large amount of noise is much less probable than a smaller amount, with an exponential fall-off:

$$P(\boldsymbol{n}) = \frac{1}{2\pi\sigma^2} e^{-\frac{\|\boldsymbol{n}\|_2^2}{2\sigma^2}}. \tag{3.12}$$

This has a straightforward motivation in other scenarios, e.g. in image restoration: there, Gaussian noise is a good model for the thermal noise generated by electronical components. Yet for us, too, it is often a good model for the type of noise we observe from point detectors.

Given the form of the Gaussian distribution's probability density function, we can take the logarithm of the problem in equation (3.11) to write the optimization problem as

$$\begin{aligned} &\max_{\hat{R},\hat{\boldsymbol{t}},\hat{K},\hat{\boldsymbol{k}}} \prod_i \frac{1}{2\pi\sigma^2} e^{-\frac{\|\hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i\|_2^2}{2\sigma^2}} \\ \equiv &\min_{\hat{R},\hat{\boldsymbol{t}},\hat{K},\hat{\boldsymbol{k}}} \sum_i \|\hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i\|_2^2. \end{aligned} \tag{3.13}$$

Therefore, in the case of Gaussian noise, the maximum likelihood estimation arises from setting $p = 2$ and $q = 2$ in equation (3.10).

3.3.2 Uniform noise

Uniform noise has a completely different nature: within its (bounded) support, all noise realizations are equally probable, and all other values occur with probability zero. As we assume that the noise is independent over various points but also over the different coordinates of a point, this volume is an axis-aligned hyperbeam, i.e., in 2D, a rectangle defined by two diagonally opposed corners $\boldsymbol{a} \rightarrow \boldsymbol{b}$. Uniform noise is applicable, among others, in problems where quantization plays an important role. In that case the support of the noise is known, but this might not always be the case. Uniform noise becomes a better model for detector noise as the average accuracy of that detector goes up (and errors tend to be the scale of a pixel or smaller), as quantization effects become significant. The pdf of uniform noise is simply given by the inverse of the interval volume inside the interval:

$$\begin{aligned} P(\boldsymbol{n}) &= \begin{cases} 1/A, \text{ when } a_d \leq n_d \leq b_d, \forall d \\ 0, \text{ elsewhere} \end{cases} \text{, where} \\ A &= \prod_d (b_d - a_d), \end{aligned} \tag{3.14}$$

and d sums over the dimensions.

For our applications, the noise interval will always be centered around the origin (because we assume the noise to be zero-mean). Using the uniform pdf in

equation (3.11), we arrive at the problem

$$\begin{aligned} &\max_{\hat{R},\hat{\boldsymbol{t}},\hat{K},\hat{\boldsymbol{k}},L} \prod_i \frac{1}{2L} \text{ subject to } \|\hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i\|_\infty \leq L, \\ \equiv &\max_{\hat{R},\hat{\boldsymbol{t}},\hat{K},\hat{\boldsymbol{k}},L} \frac{1}{L} \text{ subject to } \|\hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i\|_\infty \leq L, \end{aligned} \tag{3.15}$$

where L is the radius of the uniform noise, which is sometimes also unknown and needs to be minimized as well. In essence, we are trying to find the smallest possible noise radius that still explains all measurements (i.e. exactly allows for the worst measurement error), as that one will make the observations most likely. If we select a smaller one, the worst observation becomes statistically impossible (probability zero), and we select a larger one then all the measurements become slightly less likely as the noise can take on more different values. For more details, please refer to chapter 6.

In case L is known, any solution that lowers all reprojection errors below the threshold L is equally valid, and all others are invalid; there is nearly always a non-zero volume of optimal solutions.

If L is unknown, however, it can easily be eliminated because it can be solved in closed form in terms of the reprojections and the measurements:

$$L^\star = \|(\dots, \|\hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i\|_\infty, \dots)\|_\infty . \tag{3.16}$$

The optimization problem from equation (3.11) now becomes

$$\min_{\hat{R},\hat{\boldsymbol{t}},\hat{K},\hat{\boldsymbol{k}},L} \|(\dots, \|\hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i\|_\infty, \dots)\|_\infty, \tag{3.17}$$

from which we see that the MLE for uniform noise with unknown radius is given by equation (3.10) with $p \to \infty$ and $q \to \infty$.

3.3.3 Laplacian noise

The last form of noise we discuss is Laplacian noise. The Laplacian distribution, like the Gaussian one, models the fact that large amounts of noise are less probable than small amounts. However, in the Laplacian model, the probability doesn't diminish as fast as in the Gaussian model:

$$P(\boldsymbol{n}) = \frac{1}{2b} e^{-\frac{\|\boldsymbol{n}\|_1}{b}}, \tag{3.18}$$

as can also be seen in figure 3.2. Through a similar derivation as in section 3.3.1, one can show that the MLE in the case of Laplacian is given by using $p = 1$ and $q = 1$ in equation (3.10).

Laplacian noise mostly finds its application in the modeling of spectral coefficients, in e.g. speech recognition or image processing in the Fourier domain. In

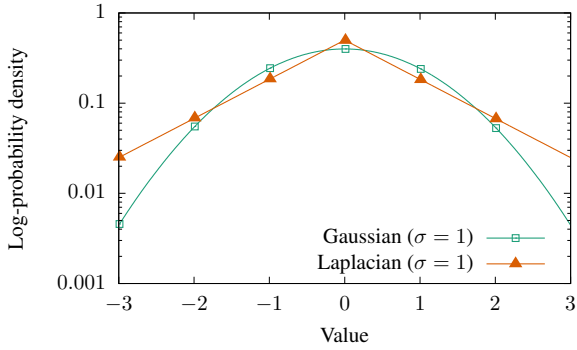


Figure 3.2: Comparison between the log-probability density functions for the Gaussian (red) and Laplacian (blue) distributions. The Laplacian noise model is much more tail-heavy: large observation noise is more likely than in the Gaussian case.

the context of object detection or tracking, and camera calibration, it is sometimes preferred because of its robustness to outliers: it is much less affected by outliers than the quadratic model that results from Gaussian noise. In the end it is much more effective to choose an approach that is able to detect and explicitly remove outliers rather than letting them influence the estimate (even slightly).

3.3.4 Selecting the noise model

As illustrated in the previous sections, the optimal choice for p and q in equation (3.10) is dictated by the used observation (noise) model; traditionally, the measurement noise is assumed to be Gaussian, and in that case the maximum-likelihood estimate is given by minimizing the error criterion with $p = q = 2$. The footnote here is that, while this is generally a valid assumption, Gaussian noise is bad at handling outliers. Remember that the optimization problem hinges on the correspondences between known 3D points on the calibration objects and their detected observations. For obvious reasons we want this calibration point detection to happen automatically; while detection methods make relatively few mistakes (see chapter 4), they do happen. An alternative that handles outliers better is ℓ_1-norm minimization, which we have shown to follow from a Laplacian noise model. However, it is bad practice to select a noise model based on its properties; the choice should be motivated by the type of noise we expect to measure. Hence, when outliers are a big concern, it is better to explicitly handle the (relatively low number of) outliers rather than choosing a robust noise model.

For that matter, ℓ_∞, where the error value is dictated by the overall worst measurement, is very much influenced by outliers. This facilitates its use to easily detect outliers, and filter them out [Olsson 2010]. After detecting and eliminating

the outliers, either manually or automatically, Gaussian noise is a valid model for the measurement noise and practical camera calibration could occur under the ℓ_2 norm.

3.4 Perspective camera calibration with the ℓ_2 norm

As noted earlier, perspective cameras are typically modelled as pinhole cameras with additional lens distortion [Sturm 2011]. Under this model, projection is separated in an extrinsic matrix (the location and orientation of the camera), an intrinsic matrix (its focal distance, skew and optical center) and the deformation coefficients (typically the Brown-Conrady "plumb bob" distortion model for radial and tangential distortion [Conrady 1919, Brown 1966]). The intrinsic matrix and the deformation coefficients are camera-dependent but typically remain constant over time, with the possible exception of the focal length in the intrinsic matrix. The extrinsic matrices, on the other hand, change from image to image. Estimating the most likely values for these entities is done by optimizing the goodness-of-fit criterion $\Phi(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}})$ from equation (3.10). Per the previous discussion, we assume Gaussian noise and hence $p = q = 2$: we minimize the mean squared errors of the ℓ_2 reprojection errors. The result is a well-behaved cost criterion:

$$\begin{aligned} \Phi(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}) &= \left\| \Big(\dots, \phi_i(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}), \dots \Big) \right\|_2 , \text{where} \\ \phi_i(\hat{R}, \hat{\boldsymbol{t}}, \hat{K}, \hat{\boldsymbol{k}}) &= \left\| \hat{\boldsymbol{u}}_i - \tilde{\boldsymbol{u}}_i \right\|_2 . \end{aligned} \tag{3.19}$$

The seminal work by Zhang et al. [Zhang 2000] still counts as the state of the art and most used camera calibration algorithm, using planar reference objects (i.e., checkerboards) for calibration, and the Matlab and C implementations by Jean-Yves Bouguet are still popular and in widespread use [Bouguet 2004]. We now give an overview of their method.

Because of the well-behaved loss function from equation (3.19), traditional non-linear optimization techniques can be used to optimize it. Yet in order to do so we first require a rough estimate of the optimal solution. Because we use planar reference objects (the calibration checkerboards), we can use the homographies between the reference object and the image planes to extract an initial estimate of the camera parameters, which is outlined in the next section. Afterwards, this initial estimate is refined with Levenberg-Marquardt optimization: the so-called bundle adjustment step. As we have already briefly noted, lens distortions are ignored in the initialization step: they are assumed to be zero. They are subsequently taken into account during the bundle adjustment procedure; initializing them to zero proves to be good enough [Bouguet 2004].

3.4.1 Homographies as initialization

In order to find an initial estimate for the camera parameters (except the distortions, whose initial guess is all zeros), we use homographies that relate one plane to another by a non-rigid transformation; this transformation can be expressed as a 3×3 transformation matrix H that acts on the homogeneous coordinates of one plane to map them onto the homogeneous coordinates of another. We will be using them to map the known ground truth plane, i.e. the known checkerboard corner positions, to the image plane, i.e. the observed checkerboard corner positions. Subsequently, we then extract an initialization of the parameters from these homographies. This algorithm is called Direct Linear Transform (DLT).

Without loss of generality, we assume that the checkerboard's ground truth 3D position is axis-aligned in the Z-plane, so that all Z-coordinates are zero. This means that we have

$$\begin{aligned} \boldsymbol{u}_i &\propto K\left[R \mid \boldsymbol{t}\right] \boldsymbol{p}_i \\ &= K\left[R \mid \boldsymbol{t}\right] \begin{bmatrix} x_i & y_i & 0 & 1 \end{bmatrix}^{\mathrm{T}} \\ &= K\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^{\mathrm{T}} \\ \begin{bmatrix} u_i & v_i & 1 \end{bmatrix}^{\mathrm{T}} &\propto H \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^{\mathrm{T}}, \end{aligned} \tag{3.20}$$

where H is the homography between the original reference points and their projections. Note that, due to the homogeneous coordinates, this homography is only defined up to a scale factor λ. Furthermore, it is clearly visible that the homography combines the effects of the intrinsic and extrinsic matrices.

Denoting the homography by its columns $H = [\boldsymbol{h}_1 \ \boldsymbol{h}_2 \ \boldsymbol{h}_3]$, we estimate it by solving the homogeneous, linear system

$$\begin{cases} u_i \boldsymbol{h}_3 \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{h}_1 \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^{\mathrm{T}} \\ v_i \boldsymbol{h}_3 \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{h}_2 \begin{bmatrix} x_i & y_i & 1 \end{bmatrix}^{\mathrm{T}} \end{cases} \quad \forall i. \tag{3.21}$$

Now, we wish to extract the intrinsic and extrinsic matrices from this homography. To do so, note that $\begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \boldsymbol{h}_3 \end{bmatrix} = \lambda K \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix}$. Because of the orthonormality of $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$, we have

$$\begin{aligned} &\boldsymbol{h}_1^T K^{-T} K^{-1} \boldsymbol{h}_2 = 0, \\ &\boldsymbol{h}_1^T K^{-T} K^{-1} \boldsymbol{h}_1 = \boldsymbol{h}_2^T K^{-T} K^{-1} \boldsymbol{h}_2, \end{aligned} \tag{3.22}$$

Which means that, per homography (per captured image of the reference object), we have two constraints on the intrinsic parameters. The intrinsic matrix has five degrees of freedom (focal length, pixel aspect ratio, skew, and optical center) but the skew is typically assumed to be zero or at least well known. This means that two different views of the calibration object (with two different homographies)

could be sufficient for camera calibration, although more views will lower the impact of measurement noise.

After solving the system equation (3.22) for the intrinsic matrix, the extrinsic parameters for each image can easily be extracted from the corresponding homography:

$$\begin{aligned} \boldsymbol{r}_1 &= \lambda K^{-1}\boldsymbol{h}_1 \\ \boldsymbol{r}_2 &= \lambda K^{-1}\boldsymbol{h}_2 \\ \boldsymbol{r}_3 &= \boldsymbol{r}_1 \times \boldsymbol{r}_2 \\ \boldsymbol{t} &= \lambda K^{-1}\boldsymbol{h}_3, \end{aligned} \tag{3.23}$$

where $\lambda = 1/\|K^{-1}\boldsymbol{h}_1\| = 1/\|K^{-1}\boldsymbol{h}_2\|$. Note that rotation matrices are part of the special orthogonal group $SO(3)$: 3×3 matrices whose rows are orthonormal and which have a positive determinant. Because of measurement noise, the extracted rotation matrices for each image do not usually lie exactly in the special orthogonal group. We simply project them orthogonally onto that group: after all, remember that this estimate only serves as a starting point for the iterative optimization.

3.4.2 Iterative improvement

In order to refine the initial guess for the camera intrinsics and extrinsics, as well as take lens distortions into account (which we assumed to be zero during the initialization), we perform Levenberg-Marquardt optimization [Levenberg 1944, Marquardt 1963]. The crux of the method is that we can analytically express the optimization of the cost function from equation (3.19) as a non-linear problem, and hence calculate the relevant Jacobians and gradients so that the Levenberg-Marquardt approach becomes viable.

In favour of brevity, a detailed formulation is omitted here; we refer to the existing publications on camera calibration [Triggs 1999, Zhang 2000] for the full mathematical derivations. However, it is worth noting that the distortions cause some trouble. As mentioned in section 2.1.3.3, lens distortions are to be applied between the extrinsic transform and the intrinsic transform but the Brown-Conrady lens distortion model is unfortunately not invertible in closed form. To this end, Heikkila et al. [Heikkila 1997] have proposed an iterative inversion procedure that is fast and robust enough to be used during the bundle adjustment step, as well as being differentiable so that we can still calculate the relevant Jacobians and gradients required by the Levenberg-Marquardt iterative optimization.

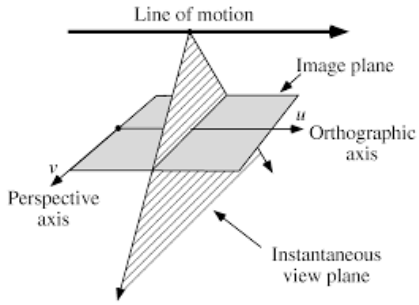


Figure 3.3: Illustration of a linear pushbroom camera. A linear sensor is moved over a scene, capturing a single image line at any given instant. In that sense, one of the directions (the perspective one) is spatial, while the other (the orthographic one) is temporal. See section 2.1.5 for more details.

3.5 LPB camera calibration with the ℓ_2 norm

So far, this chapter has only discussed the calibration of perspective cameras. However, for other camera models we obviously cannot simply adopt the perspective camera calibration procedure. As the linear pushbroom camera reappears later in this dissertation, and because we have found issues with the existing calibration techniques and proposed a more robust method ourselves, we discuss the calibration of their parameters here. Linear pushbroom cameras are a mix of orthogonal and perspective cameras, as discussed in chapter 2. Figure 3.3 refreshes the memory: a one-dimensional sensor line is swept over a scene; in the resulting image, the X-dimension is the dimension of the sensor line, with perspective projection, while the Y-dimension is the temporal dimension, performing orthogonal projection over time (although we will simply interpret this dimension as a spatial one). This camera model occurs in two specific scenarios: in imaging satellites, and in hyperspectral cameras based on the diffraction concept. Due to the rarity of this type of camera, the corresponding calibration literature is not very extensive.

The seminal work on linear pushbroom cameras is focused on satellite applications [Gupta 1997]. In that case, the model is only an approximation: while the satellites are in an orbit around our planet (which is, indeed, not flat!), sufficiently short parts of their trajectories are approximately straight. Yet for other applications, such as the hyperspectral scanner we discuss below, the model is not an approximation. More of an issue, however, is that the authors of [Gupta 1997] rely on known correspondences between 3D ground reference points and 2D camera observations for the calibration of the cameras: they use manual annotations of well-known landmarks visible to the satellites – a method that scales poorly when high accuracy and therefore many observations are needed. Such manual annotation may be feasible in satellite applications, where it carries a relatively small

relative cost, but they are not plausible in more resource-constrained scenarios. While several other works have performed automatic calibration by utilizing an auxiliary perspective camera [Hui 2013, Sun 2016, Li 2016] to obtain the requisite correspondences, Draréni et al. have instead introduced the use of checkerboards for linear pushbroom calibration [Draréni 2011]. By using a traditional checkerboard, they are able to easily generate large numbers of point correspondences, improving the accuracy of the calibration. And as the hyperspectral cameras tend to image wavelengths around the visual spectrum, which behave the same way as visible light, the calibration images are similar and the checkerboard detection algorithms discussed in chapter 4 can be used.

Similar to the calibration technique for perspective cameras, we first find an initial estimate for the camera parameters (again using homographies) and subsequently refine it using iterative optimization techniques. However, the method by Draréni et al. cannot handle specific – but common – orientations of the observed checkerboard: among others, the method becomes numerically unstable when the checkerboard is parallel to the imaging plane. In order to understand this problem, we first introduce the method by Draréni et al. and afterwards outline our proposed solution to these issues.

3.5.1 The calibration problem

Recall from section 2.1.5 that a point $\boldsymbol{p} = [x\ y\ z\ 1]^{\mathrm{T}}$ in the world coordinate system is transformed to a point $\boldsymbol{p}_c = [x_c\ y_c\ z_c\ 1]^{\mathrm{T}}$ in the camera coordinate system, and subsequently projected to the image plane as $\boldsymbol{u} = [u\ v\ 1]^{\mathrm{T}}$:

$$\begin{bmatrix} x_c & y_c & z_c & 1 \end{bmatrix}^{\mathrm{T}} = [R \mid \boldsymbol{t}] \begin{bmatrix} x & y & z & 1 \end{bmatrix}^{\mathrm{T}}$$
$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \propto \underbrace{\begin{bmatrix} f & 0 & u_0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} x_c \\ y_c z_c \\ z_c \end{bmatrix} \tag{3.24}$$

We now wish to estimate the unknown focal distance f, optical center u_0, scanning speed s, as well as the transformation matrices $[R \mid \boldsymbol{t}]$ for all cameras, based on the measurements $[u\ v]^{\mathrm{T}}$ for known 3D points $[x\ y\ z]^{\mathrm{T}}$. As one can see, there is now a non-linear term in this projection equation, as we require $y_c z_c$. This becomes even more evident when writing down the equations required to perform the camera calibration. Similar to the perspective camera calibration, we will assume that the reference object is axis-aligned inside the Z-plane, with the

relevant points at locations $\begin{bmatrix} a_i & b_i & 0 \end{bmatrix}^{\mathrm{T}}$. Formulated mathematically, we get

$$\begin{aligned} \begin{bmatrix} x_c & y_c & z_c & 1 \end{bmatrix}^{\mathrm{T}} &= \begin{bmatrix} R|\boldsymbol{t} \end{bmatrix} \begin{bmatrix} a_i & b_i & 0 & 1 \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix} \begin{bmatrix} a_i & b_i & 1 \end{bmatrix}^{\mathrm{T}} \\ &= \begin{bmatrix} a_i r_{11} + b_i r_{12} + t_1 \\ a_i r_{21} + b_i r_{22} + t_2 \\ a_i r_{31} + b_i r_{32} + t_3 \end{bmatrix} \end{aligned} \tag{3.25}$$

We can now no longer write down the projection equation as a single linear equation such as in equation (3.7), which complicates further steps:

$$\begin{aligned} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &\propto \underbrace{\begin{bmatrix} f & 0 & u_0 \\ 0 & s & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{K} \begin{bmatrix} x_c \\ y_c z_c \\ z_c \end{bmatrix} \\ &= K \begin{bmatrix} a_i r_{11} + b_i r_{12} + t_1 \\ (a_i r_{21} + b_i r_{22} + t_2)(a_i r_{31} + b_i r_{32} + t_3) \\ a_i r_{31} + b_i r_{32} + t_3 \end{bmatrix} \end{aligned} \tag{3.26}$$

We would rather be able to express the relationship between the observed 2D projections and the known 3D locations as a linear system. As the locations of the reference points in the world coordinate system are perfectly known anyway, we instead adopt the Veronese mapping [Barreto 2006], which lifts the homogeneous location $\begin{bmatrix} a_i & b_i & 1 \end{bmatrix}^{\mathrm{T}}$ to the sixth dimension by extending it with $\begin{bmatrix} a_i^2 & b_i^2 & a_i b_i \end{bmatrix}^{\mathrm{T}}$. As a result, the relationship between the Veronese mapping of the reference point locations and their projection becomes linear again:

$$\begin{aligned} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &\propto K \underbrace{\begin{bmatrix} r_{11} & r_{12} & t_1 \\ r_{21} t_3 + r_{31} t_2 & r_{22} t_3 + r_{32} t_2 & t_2 t_3 \\ r_{31} & r_{32} & t_3 \end{bmatrix}}_{P_1} \begin{bmatrix} a_i \\ b_i \\ 1 \end{bmatrix} \\ &\quad + K \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ r_{21} r_{31} & r_{22} r_{32} & r_{21} r_{32} + r_{22} r_{31} \\ 0 & 0 & 0 \end{bmatrix}}_{P_2} \begin{bmatrix} a_i^2 \\ b_i^2 \\ a_i b_i \end{bmatrix} \end{aligned} \tag{3.27}$$

Now, similar to the perspective case, we have a homography-like matrix $H = K \begin{bmatrix} P_1 & P_2 \end{bmatrix}$, which has 12 non-zero entries:

$$H = \frac{1}{t_3}\begin{bmatrix} fr_{11}+u_0r_{31} & fr_{12}+u_0r_{32} & ft_1+u_0t_3 \\ s(r_{21}t_3+r_{31}t_2) & s(r_{22}t_3+r_{32}t_2) & st_2t_3 \\ r_{31} & r_{32} & t_3 \\ 0 & 0 & 0 \\ sr_{21}r_{31} & sr_{22}r_{32} & s(r_{21}r_{32}+r_{22}r_{31}) \\ 0 & 0 & 0 \end{bmatrix} \tag{3.28}$$

In Draréni et al.'s original publication, there was no scaling of the homography matrix with t_3. However, as it makes notations easier and – more importantly – as $t_3 \neq 0$ because that would imply the physical impossibility of the camera being at the same location as the reference object, we instead scale the homography matrices so that $h_{33} = 1$. By multiplying equation (3.27) with the cross-product skew matrix of $\begin{bmatrix} u & v & 1 \end{bmatrix}^{\mathrm{T}}$, Draréni et al. obtain a linear homogeneous system in H's entries:

$$\begin{bmatrix} a_i & b_i & 1 & 0 & 0 & 0 & 0 & 0 & 0 & -a_iu & -b_iu & -u \\ 0 & 0 & 0 & a_i & b_i & 1 & a_i^2 & b_i^2 & a_ib_i & -a_iv & -b_iv & -v \\ -a_iv & -b_iv & -v & a_iu & b_iu & u & a_i^2u & b_i^2u & a_ib_iu & 0 & 0 & 0 \end{bmatrix} \boldsymbol{h} = \mathbf{0} \tag{3.29}$$

In this system, $\boldsymbol{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{24}, h_{25}, h_{26}, h_{31}, h_{32}, h_{33}]^{\mathrm{T}}$ contains the entries of H in a single vector. This homogeneous system is solved by singular value decomposition to yield an estimate of H, and from this estimate, we now need to extract an initial estimate of the intrinsic and extrinsic parameters.

3.5.2 Extracting the camera parameters from the homography

Similar to equation (3.22) in the perspective camera case, we leverage the orthonormality of the rotation matrix's columns to obtain constraints for the intrinsic parameters. As we have mentioned before, rotation matrices are special orthogonal matrices: their rows are orthonormal, and their determinant equals 1.

This yields us the constraints that

$$\begin{aligned} \boldsymbol{r}_1^{\mathrm{T}}\boldsymbol{r}_1 &= \|\boldsymbol{r}_1\|_2^2 = 1, \\ \boldsymbol{r}_2^{\mathrm{T}}\boldsymbol{r}_2 &= \|\boldsymbol{r}_2\|_2^2 = 1, \\ \boldsymbol{r}_1^{\mathrm{T}}\boldsymbol{r}_2 &= \boldsymbol{r}_1^{\mathrm{T}}\boldsymbol{r}_2 = 0. \end{aligned} \tag{3.30}$$

In order to leverage these constraints for extracting the camera parameters, Draréni et al. express the first two columns of the rotation matrix as:

$$\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 \end{bmatrix} = \frac{t_3}{sf} LM, \tag{3.31}$$

where

$$\begin{aligned} L &= \begin{bmatrix} s & 0 & -su_0 \\ 0 & f/t_3 & 0 \\ 0 & 0 & sf \end{bmatrix} \\ M &= \begin{bmatrix} h_{11} & h_{12} \\ h_{24}/h_{31} & h_{25}/h_{32} \\ h_{31} & h_{32} \end{bmatrix} \end{aligned} \tag{3.32}$$

However, this entails dividing by entries of the rotation matrix, r_{31} and r_{32}. As these entries can very well be zero, this results in numerical instabilities for those cases. Among others, these are close to zero for checkerboards that are nearly parallel to the XY plane, i.e. parallel to the image plane of the camera. Therefore one would have to take pains to avoid this situation in practical calibration problem, or weed out such positions.

In our case, see chapter 9, we had no control over the calibration set-up. The checkerboards were mounted on a conveyor belt system, and they were *always* roughly parallel to the image plane. In order to solve the numerical issues that result, we have proposed the following formulation for M, which is an alternative way of extraction the relevant entities from the estimated homography matrix H:

$$\begin{aligned} M &= \begin{bmatrix} h_{11} & h_{12} \\ \dfrac{h_{33}h_{21} - h_{31}h_{23}}{h_{33}^2} & \dfrac{h_{33}h_{22} - h_{32}h_{23}}{h_{33}^2} \\ h_{31} & h_{32} \end{bmatrix} \\ &= \begin{bmatrix} h_{11} & h_{12} \\ h_{21} - h_{31}h_{23} & h_{22} - h_{32}h_{23} \\ h_{31} & h_{32} \end{bmatrix}, \end{aligned} \tag{3.33}$$

where the second step takes into account that $h_{33} = 1$, which simplifies the expression. We now divide by the z-offset of the translation vector, rather than by entries of the rotation matrix. As mentioned earlier, this z-offset is never close to zero, as that would mean that the reference object resides in roughly the same location as the camera lens; a physical impossibility that would result in bigger problems than not being able to calibrate the camera. One could simply make sure that checkerboards are never quasi-parallel to the image plane, but in some cases (e.g. the BAHAMAS project) this is not possible, and we show below that this is an unnecessary, artificial restriction of [Draréni 2011].

Using this novel expression for M, we now formulate the orthonormality constraints of the rotation matrix as follows, reintroducing the image subscript c:

$$\begin{cases} (M_c^{\mathrm{T}} L_c^{\mathrm{T}} L_c M_c)_{12} = (M_c^{\mathrm{T}} L_c^{\mathrm{T}} L_c M_c)_{21} = 0 \\ (M_c^{\mathrm{T}} L_c^{\mathrm{T}} L_c M_c)_{11} = (M_c^{\mathrm{T}} L_c^{\mathrm{T}} L_c M_c)_{22} \end{cases} \tag{3.34}$$

We call

$$X_c = L_c^{\mathrm{T}} L_c = \begin{bmatrix} s^2 & 0 & -s^2 u_0 \\ 0 & f^2/t_{c,3}^2 & 0 \\ -s^2 u_0 & 0 & s^2(u_0^2 + f^2) \end{bmatrix} = \begin{bmatrix} x_1 & 0 & x_2 \\ 0 & x_{c,4} & 0 \\ x_2 & 0 & x_3 \end{bmatrix}, \tag{3.35}$$

of which only the center element changes between the different images – the other entries depend solely on the intrinsic parameters. By collecting the orthogonality constraints for each of the C calibration images, we have a homogeneous system with $2C$ equations in $3 + C$ parameters. After solving this system up to scale, we have estimates for u_0 and f:

$$u_0 = -x_2/x_1 \qquad \text{and} \qquad f = \sqrt{\frac{x_3}{x_1} - u_0^2}. \tag{3.36}$$

Using only the second equations from equation (3.34), we obtain a set of equations that can be solved for $t_{c,3}$ and s. As the system actually entails solving for $1/t_{c,3}^2$ and $1/s^2$, the sign of the entities is still unclear. As $t_{c,3}$ has a geometrical meaning (it is the signed distance from the camera center to the calibration objects), it is necessarily positive – this is also called the cheirality constraint [Hartley 1993]. Inverting the sign of S is equivalent to inverting the Y-axes of all camera coordinate systems - the choice is therefore arbitrary, and we choose it to be positive.

At this point, it is straightforward to extract the values of the missing extrinsic parameters from the homography in equation (3.28). Afterwards, we project the rotation matrix estimates on the nearest $SO(3)$ matrices, and perform bundle adjustment. While the precise formulations of the Jacobians and gradients differ from that of pinhole cameras, the same principles from section 3.4.2 remain valid – see [Draréni 2011].

3.5.3 Comparison with the existing method

We now show that, aside from being able to handle checkerboard orientations which are numerically unstable for Draréni et al.'s approach, our proposed method also performs better for the other cases [Draréni 2011]. To do so, we have evaluated on both a synthetic dataset (similar to the one originally used by Draréni et al.) and a real one (as captured by us in the context of the BAHAMAS project). In the real dataset, Draréni et al. is unable to produce any satisfactory results, as the planes are nearly parallel to the image plane and the method becomes numerically unstable because of the division by rotation matrix elements; in contrast, our proposed method has no problem yielding good results. Even on the synthetic

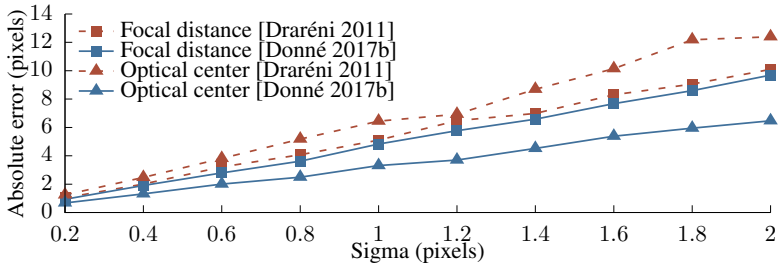


Figure 3.4: Calibration errors w.r.t. the noise level in the point measurements.

dataset, which did not contain nearly-parallel planes, our method outperforms that of Draréni et al., indicating that our formulation is more robust even when no degenerate positions plague the existing method.

3.5.3.1 Synthetic data

We perform several tests of our proposed algorithm using synthetic data, with the same settings as [Draréni 2011]. A planar calibration grid of 10×10 corners is captured by a virtual camera with a 1000×1000 image resolution, $f = 1000$ and $u_0 = 500$. The scanning speed is 50 pixels (which means that for each subsequent line of the image, the camera has moved a distance equivalent to 50 times the pixel width). As mentioned before, this dataset does not contain nearly-parallel planes and the aforementioned numerical instability does not occur for Draréni's method.

First of all, we evaluate the robustness to noise. For 10 planes oriented randomly, we pollute the measurements with additive white Gaussian noise of varying variance. We perform 1000 independent calibration tests for each of several σ values between 0.2 and 2.0 pixels (an interval of realistic values) and report the average absolute errors for both the focal length and the optical center, as in [Draréni 2011]. We see in figure 3.4 that the error increases roughly linearly with rising σ. While our results for the Draréni method match those reported in [Draréni 2011], our proposed method outperforms their method and finds better solutions.

Secondly, we evaluate the performance of the methods with respect to the amount of data available, i.e. the number of calibration images. As figure 3.5 shows, the accuracy increases notably up to about 20 planes, after which the marginal gain levels off. Similar to the noise perturbation experiment, we see that our proposed method consistently performs better than the existing approach.

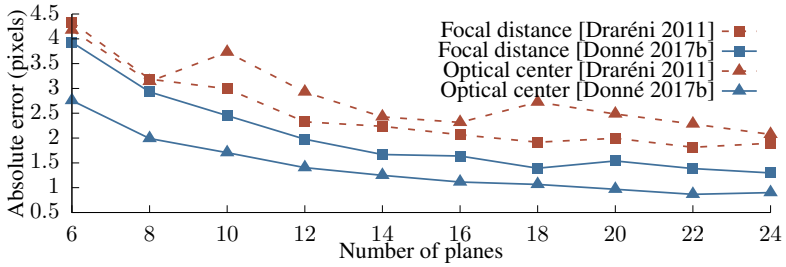


Figure 3.5: Calibration errors w.r.t. the number of measurements available. We attribute the relatively bigger impact on the optical center accuracy to the absence of the square root in equation (3.36).

Figure 3.6: An overview of the automatic green-house set-up. A conveyor belt system controls the movement of the plants between feeding stations, growing areas and the photo room. In the photo room, several RGB cameras as well as a long-wave infrared and a hyperspectral short-wave and near-visual infrared cameras image the plants periodically. The hyperspectral camera follows the linear pushbroom model and requires special calibration algorithms. See chapter 9 for more details on the full project.

3.5.3.2 Experimental set-up

For the real-world evaluation of our proposed method, we calibrate a hyperspectral Specim camera. This camera is inside an imaging cabin controlled by a robotics system installed at the Flemish Institute for Biotechnology in Ghent (figure 3.6).

The conveyor belt system is only able to raise or lower objects, or rotate them in the horizontal plane. Therefore, as the patterns are mounted horizontally on the system and the camera is a top-down view, the planar patterns are always roughly parallel to the image plane.

We wish to find the accurate position of the camera with respect to the robot. For this, we have access to some manufacturer-specified knowledge: the camera has a focal distance of 500 pixels, and the optical center of the sensor line is at 160 pixels. We also know the offsets of the robot between various scans: we perform four calibration scans at two different heights, with a displacement of $200mm$

(roughly 60% of the checkerboard size). Using this information to specify the initial estimate, and subsequently optimizing the estimate with bundle adjustment, we can estimate the camera position with respect to the conveyor belt system. After refinement, the estimated height difference between the scans is 198.81 mm, an error of less than 1% with the ground truth robot positions.

This calibration allowed us to accurately relate 3D reconstructions built from other cameras to the hyperspectral scans, as further outlined in chapter 9. In contrast, the method proposed by Draréni et al. could not calibrate this set-up at all, yielding nonsense results due to numerical instability discussed earlier.

3.6 Contributions

The field of camera calibration is very mature, and for the common camera models the techniques function well. However, as different types of cameras become more readily available, these also require calibration. In this chapter we have discussed how to approach the calibration of a camera system, both for a traditional perspective camera and for the linear pushbroom camera of a hyperspectral system. We have outlined the general optimization procedure for camera calibration, outlined various measurement noise models, and discussed the calibration for a Gaussian noise model in detail. After illustrating the drawbacks of the pre-existing calibration methods for linear pushbroom cameras, we have outlined our published method for alleviating these issues. As an additional benefit from adapting the existing technique to be able to handle previously degenerate checkerboard positions, our proposed technique is also more robust in the presence of noise and more accurate for the same number of measurements. The work discussed in this chapter has led to one peer-reviewed publication:

- *Robust Plane-based Calibration for linear cameras*, Simon Donné, Hiep Luong, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé, Bart Goossens and Wilfried Philips. Proceedings of the International Conference on Image Processing, ICIP 2017 [Donné 2017b].

4

Detecting calibration objects

" Now is no time to think of what you do not have. Think of what you can do with that there is. "

— Ernest Hemingway, *The Old Man and the Sea*

In chapter 3, we have discussed how to estimate the parameters for the camera models introduced in chapter 2. To do so, camera calibration techniques use the measured 2D projections of known 3D locations to estimate the camera parameters that best explain these measurements. The current chapter discusses how to get the corresponding measurements for known 3D locations: in order to relate image measurements to the world coordinate system, we need to know which observation corresponds to which known 3D location. To this end, we can use a variety calibration objects: objects that we can easily detect and locate in an image, and for which we know the 3D shape accurately. It is possible, but relatively hard and inaccurate, to estimate camera parameters based only on the scene itself, without any explicitly added calibration objects; that is what Structure-from-Motion (SfM) and Simultaneous Localization and Mapping (SLAM) techniques do. However, aside from the inevitable outliers they face in their detection and matching, they also assume that only the camera extrinsics are unknown and that we already accurately know the lens deformation model and intrinsic matrix; these distortion coefficients and intrinsic parameters still need to be estimated beforehand. As such, calibration objects still play a valuable role in the calibration of a camera system, provided that they can be detected and matched easily.

First, we give an overview of the most popular calibration objects, each with

their own advantages and drawbacks. While these objects and patterns are designed as to be easily distinguishable from the rest of the captured scene, automating this task is not necessarily easy or straightforward. Therefore, we discuss their automatic detection, focusing on 2D checkerboards. Checkerboards are the most-used calibration pattern in computer vision, being both easy to construct and easy to detect, as well as yielding many 2D-to-3D correspondences per captured image. In the course of this doctoral work, we have proposed and evaluated the use of deep learning for automatically detecting checkerboard corners [Donné 2016b], and this chapter also discusses that contribution and its performance compared to other state-of-the-art methods for checkerboard corner detection. We illustrate that, in addition to matching or improving on the existing methods for checkerboard corner detection, the network can easily be retrained for a novel type of calibration pattern which, with a traditional approach, would otherwise require a manually designed feature detection algorithm.

4.1 Calibration objects

As we have mentioned in the introduction, calibration objects should fulfill two roles: they should be easily detected and located in the captured images, and they should provide accurate 3D to 2D correspondences, i.e. they should have a well-known 3D shape. We now discuss four groups of calibration objects, based on their dimensionality: 0D spheres, 1D poles, 2D checkerboards and 3D calibration objects. Generally speaking, the number of corresponding points increases as the dimensionality of the calibration object increases, but so does the difficulty in constructing them and detecting the objects in an image. As it is by far the most-used computer vision calibration object, we only discuss the detection of checkerboards in detail in this dissertation; we leave the discussion on the detection of the other objects to the referenced publications. To close this section, we also briefly discuss the need for application-specific calibration patterns, such as the perforated metal grid we use for thermal camera calibration in chapter 9.

4.1.1 Spheres

Perhaps the most basic calibration object is a sphere [Daucher 1994]: small, single points are not really distinguishable on the images, and the brightly colored dots used for, e.g., motion-capture are large enough to be labeled spheres. We assume that the diameter of the sphere is known, but if it is not, we could just say the sphere has a diameter of 1 arbitrary unit: the resulting reference system will only differ from the real world by an unknown scale factor R, which is the actual radius of the sphere. In case the intrinsic matrix of the camera is already known, a single sphere's silhouette and its radius can be used to estimate the camera's distance to

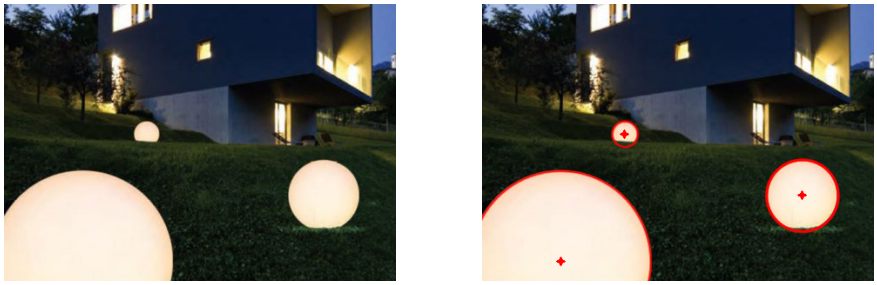


Figure 4.1: Glowing spheres in the scene can be easily detected (often, a simple threshold will do), after which the projection of their center can be estimated as the center of the sphere silhouette to yield a point observation that corresponds with the known 3D location.

this sphere [Guan 2015], which is useful for the extrinsic calibration of cameras that were previously intrinsically calibrated.

For automatic detection, self-lit spheres or brightly colored spheres are used (illustrated in figure 4.1). This means that they can be easily detected in any setting: often, a simple threshold will do. In general, a 3D sphere will be projected onto the image as an ellipse. The projected center of the sphere can be approximated as the center of its projection, and this can be coupled to a well-known 3D point of the sphere: its center. As such, each image of sphere only yields us one correspondence between a well-known 3D location and a 2D point measurement: we need many images for an accurate camera calibration.

Furthermore, spheres have the distinct advantage that they have a consistent silhouette which does not depend on the vantage point, as well as a highly regular shape. This is particularly important when calibrating wide-spread camera networks, that completely surround the object. The observation of other objects, like the checkerboard discussed later, can change drastically depending on the viewing angle. Specifically for the checkerboard, it goes so far as to disappear when viewed from the side, as it is a flat 2D object. Additionally, spheres can be detected and calibrated with, even if the sphere is only partially visible [Agrawal 2003]. For this reason, spheres have gained some popularity for extrinsically calibrating camera networks [Zhang 2007, Guan 2015].

4.1.2 Ranging poles

One step up from a simple sphere, we have the ranging poles. Rather than having only a single sphere with a given radius, we now have a colored rod of known length, possibly with differently colored segments. Similar to the calibration spheres, ranging poles exhibit roughly the same appearance regardless of the viewing angle, subject only to foreshortening. The pole consists of several segments of known length in distinct colours, so that their borders are easily detected and linked to

Figure 4.2: Ranging poles are a one-dimensional option for camera calibration, which are mainly popular for their ease of handling at larger scales (several meters long) in mobile mapping and land surveying activities.

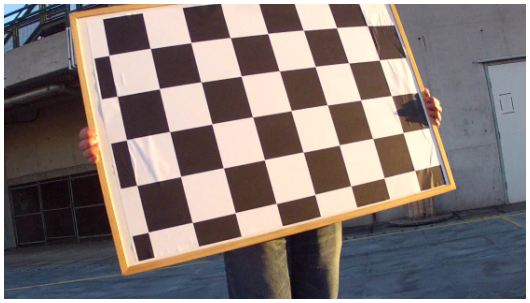


Figure 4.3: Example of the classic calibration checkerboard: a large rectangular, black-and-white checkerboard pattern.

their known 3D coordinates [Steger 1998, Lehtomäki 2010]. When we observe the poles in sufficient different positions, we have in effect observed a 3D point cloud, which can be used for calibration [Zhang 2004, Wu 2005]. Ranging poles are popular in mobile mapping applications [Douterloigne 2013], as well as for use by land surveyors (see figure 4.2) because they can easily be created on the scales required for these applications: creating such large spheres or checkerboards, or working with them, is not practical.

Compared to simple spheres, ranging poles yield significantly more correspondence points per captured image while still being easy to detect, assuming that the colors are sufficiently distinct from those naturally occurring in the scene.

4.1.3 Checkerboards

The main advantage of the calibration spheres and ranging poles is that they are very easy to find in the image. However, the number of observations per captured

Figure 4.4: The checkerboard clearly shows the camera distortion: what should have been the straight lines of the checkerboard are projected as decidedly non-linear curves. This image was taken with a GoPro, a popular consumer camera, which is known for its large distortions.

image is rather low: one per sphere, and only a handful per ranging pole. So while these objects are easy to detect, they provide only a small amount of information per captured image. Checkerboards, shown in figure 4.3 fulfill both conditions: they comprise a highly regular texture not seen in nature (and hence easy to detect) and they provide a large number of easily detectable points per captured image, with all of the black-and-white corners. Their main drawback is that they are not visible from all angles: they disappear when viewed from the sides, and it becomes progressively harder to detect and accurately locate the checkerboard corners for oblique observation angles. Despite this, they are by far the most used calibration object due to their ease of use and their distinctiveness, as well as the ease of construction at the various scales computer vision scientists need: from millimeter-size for macro-photography applications to meter-size for outdoor scenarios.

Given an observation of a checkerboard, we are only able to correctly link the observations to the correct 3D corners up to a rotation of 180 degrees (assuming a rectangular rather than a square layout, which is typically the case) as is shown in figure 4.5. By fixing the maximum rotation of the camera, or equivalently the checkerboard (see section 3.1 on the ambiguity between the world reference system and the camera reference systems) to less than that, we have the corner correspondences between images without any ambiguity. Because they typically fill a large portion of the image with what should be straight lines, they also offer a visual cue for the distortion by the camera lens as illustrated in figure 4.4.

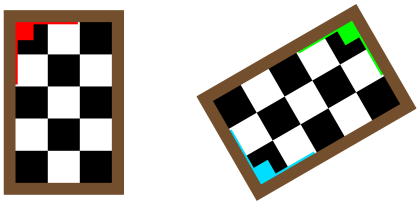


Figure 4.5: When the checkerboard rotates, we do not know for sure whether the red corner ended up at the green location or at the blue location. By restricting the maximum checkerboard rotation to less than 180 degrees, we can be sure that it is the green one.

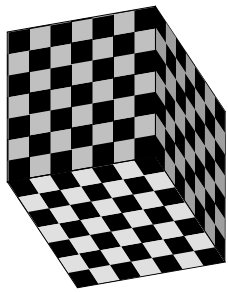


Figure 4.6: Three checkerboards perpendicular to each other to form a 3D calibration object.

4.1.4 3D calibration objects

Given that checkerboards have proven so successful, it seems natural to examine the use of 3D dimensional calibration objects, which could provide even more correspondences per image. Three-dimensional calibration patterns have the advantage that we could ostensibly perform the full camera calibration from a single image, as we have the corresponding observations for a large 3D cloud of points.

This is, for example, applicable in the calibration of satellite cameras. For these cameras, constructing calibration objects that can be observed by orbiting satellites is impractical at best. Therefore, they rely on manually annotated landmarks: points on the surface of the earth with accurately known coordinates. In essence, the entire earth serves as a 3D calibration object in this case.

For such an application this may be feasible, as there are relatively few satellites and the marginal cost of the manual annotation is low. However, for more general applications with more common camera models, such manual annotation is unfeasible. If one still wishes to perform calibration from a single camera image, it is possible to use 3D calibration objects, such as illustrated in figure 4.6. Based on the 3D-2D correspondences, the camera can be calibrated by solving a single homogeneous linear system with a method called the Direct Linear Transform (DLT). However, practical usage is hindered by the required accuracy for both the 3D rig construction and the measured locations for the 2D points; due to the nature of the method, numerical instability can occur if either of these is lacking [Forsyth 2011]. The conclusion is that, usually, one requires multiple captures of the calibration rig anyway, both for numerical stability of the resulting system and for covering the entire image with the observed points. Hence there is no convincing advantage to use the 3D objects, which are both harder to construct and often less practical to handle, so that the traditional flat checkerboards carry most researchers' preference.

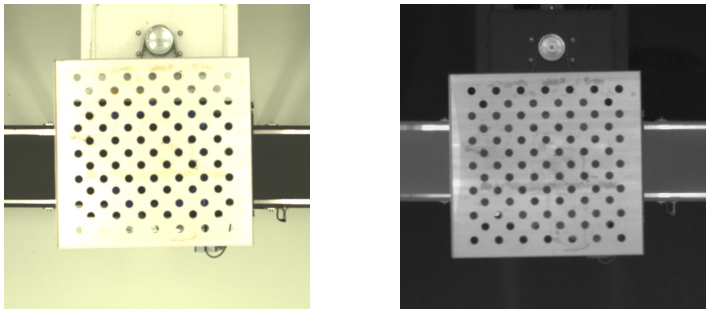


Figure 4.7: The metal grate used for calibrating the RGB (left) and LWIR (right) cameras with respect to one another.

4.1.5 Application-specific objects

There are scenarios in which classic calibration objects are unusable. As mentioned earlier in this chapter, this is the case for satellite camera calibration simply due to the sheer scale problem, with calibration objects needing to be large enough to be resolved from outer space. Another such application is the case of non-visual spectrum cameras. Specifically in the case of thermal cameras (imaging in the long-wave infrared LWIR domain), black and white printed patterns are indistinguishable on the captured image, as the thermal contrast is negligible.

We encountered this problem in the BAHAMAS ICON project, a project in cooperation with the VIB Center for Plant Systems Biology in Ghent. There, we wished to calibrate the camera set-up of an imaging chamber containing both visual-spectrum cameras and a long-wave infrared (LWIR) thermal camera. In order to estimate the relationship between the LWIR camera and the visual spectrum cameras, we required a calibration object that could be easily detected and whose points could be accurately located. To this end, we have used the perforated metal plate shown in figure 4.7. After warming it in the sun for a while, it is easily distinguished in the thermal domain, while it is, by virtue of its construction, highly reflective and appears very bright in the visual domain because of the lighting. As a result, we can easily detect the center of the grating holes by finding the local minima in a blurred version of the images. Those detected positions are shown in figure 4.8.

4.2 Detecting checkerboards

We now discuss specifically the detection of 2D checkerboards in the images. We restrict ourselves to the detection of checkerboards, which are by far the most popular and most used calibration object. Several techniques exist for the detection of

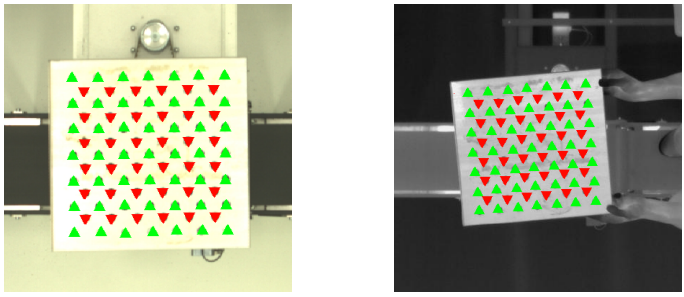


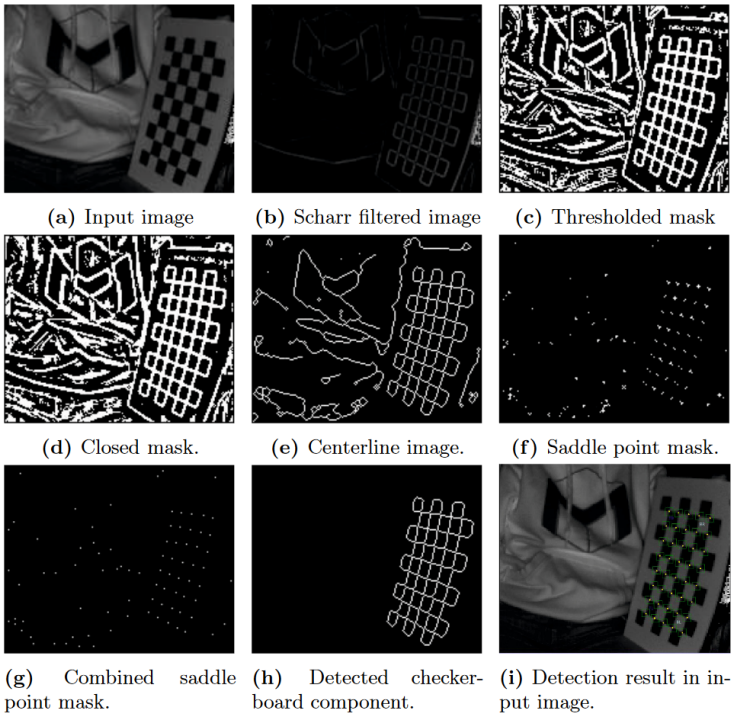
Figure 4.8: The detected grating centers for the intermodality calibration. These were found by manually annotating the four corner points, spawning the right grid, and then simply doing gradient descent on the blurred intensity images. As the metal plate appears very bright in both images, the grating hole centers are indeed located in local minima.

checkerboard features, as well as for constructing the checkerboard structures out of them. We first discuss the state-of-the-art, and then highlight how our proposed deep-learning approach fills a gap in literature by providing a quick and robust technique that is easily adapted to different calibration objects. As we leverage deep learning techniques to achieve this result, we include a brief introduction to deep learning and its concepts before outlining our approach, for readers who may be unfamiliar with deep learning techniques.

4.2.1 Literature study

Initially, checkerboard detection relied on general feature detectors such as the Harris corner detector [Harris 1988], or SUSAN (Smallest Univalue Segment Assimilating Nucleus) [Smith 1997, Zhu 2009] or Moravec [Moravec 1977] features. A more complex combination of general image features was proposed by Placht et al. with ROCHADE: Robust Checkerboard Advanced Detection [Placht 2014]. The centerlines of a thresholded Scharr-filtering of the input image are calculated and used to compute the saddle-points, which are the detected corners – this is illustrated in figure 4.9.

Alternatively, under the assumption that lens distortion is negligible, the checkerboard can be detected as two sets of lines converging on different vanishing points, which can be detected using the Hough transform [Wang 2007, De la Escalera 2010]. However, it is rare for the lens not to distort the lines at all; such the Hough transform method is only really applicable when the lens distortion is already accurately known and can be corrected for. However, the estimation of these distortion parameters is quite often one of the goals of the calibration procedure; to escape this catch-22 situation, the template extraction algorithm itself should be robust against such lens distortions. Therefore, the Hough transform is rarely applicable.

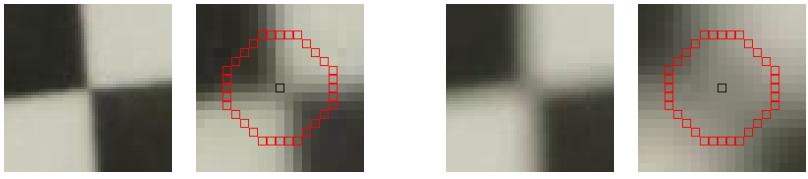


(a) Input image (b) Scharr filtered image (c) Thresholded mask

(d) Closed mask. (e) Centerline image. (f) Saddle point mask.

(g) Combined saddle point mask. (h) Detected checkerboard component. (i) Detection result in input image.

Figure 4.9: The various steps involved in the ROCHADE checkerboard detection. Image taken directly from [Placht 2014].



(a) **(b)**

*Figure 4.10: Illustration of the effect of focal blur on the corners. (**a**) in-focus corner, with the six-radius circular boundary highlighted on a detailed version; (**b**) badly out-of-focus corner with the similar enlarged version.*

However, given the distinct nature of the checkerboard pattern, more targeted approaches yield better results. For one, using a histogram-based approach, it is possible to start off with a rough detection of the areas in the image likely to contain a checkerboard [Arca 2005, Su 2013], which can already speed up the application of the above feature detectors dramatically. In the open-source computer vision library OpenCV, the checkerboard detection method is an algorithm by Vezhnevets [Vezhnevets 2005], which operates by detecting black quadrangles in the image and combining those into checkerboards. This approach was extended in OCamCalib [Rufli 2008] with a better checkerboard construction algorithm and preprocessing steps to handle blurred or distorted images.

Recently, features have been specifically designed for checkerboard feature detection [Arca 2005, Zhu 2009, Ha 2009, Bennett 2014, Bok 2016]. In [Ha 2009, Bennett 2014, Bok 2016], the authors consider circular neighborhoods of the corner candidates (see also figure 4.10): the intensities of the circular boundary form the corner candidate's feature vector. The authors of [Ha 2009, Bok 2016] are able to select the corner points of the checkerboards by counting and scoring the sign changes over this circular boundary: feature corners sport four distinct intensity steps over the feature vector, or a characteristic path. With ChESS (Chessboard Extraction by Subtraction and Summation) [Bennett 2014], it is shown that while such an approach works well, it may result in many false positives, and a more complex criterion is formulated based on the circular boundaries and the correlations of its phase-shifted versions. Bennett et al. go on to discuss the various false positives and extend their ChESS feature to account for these [Bennett 2014]. The main drawback is that the technique has been tailored for a low degree of lens distortion: the detection assumes orthogonal angles in the checkerboard quadrangles—a valid assumption when the checkerboard is parallel to the imaging plane, but less practical in a multi-camera set-up where it is no longer possible to have the checkerboard be parallel to all observing image planes. In [Arca 2005, Zhu 2009], the complete local neighborhood of corner candidates is considered, rather than only a circular boundary. Zhu et al. [Zhu 2009] match circular corner templates to the local neighbourhoods, while Arca et al. [Arca 2005] divide the neighborhood into nine sectors, a center and eight sectors, whose statistics are compared against hard-coded rules for corner detection.

As we aimed to apply deep learning to this problem, it is also valuable to discuss the application of machine learning to pixel-wise processing and feature detection in general. Recently, deep learning approaches have proven quite effective at pixel-wise processing: the super-resolution networks SRCNN [Dong 2014] and its faster variant F-SRCNN [Dong 2016] illustrate that even a few well-trained layers can be extremely effective for pixel-wise processing. Notably, the FAST (Features from Accelerated Segment Test) [Rosten 2006] image corner detector is built upon machine learning foundations, as is its successor, FAST-ER (FAST -

Enhanced Repeatability) [Rosten 2010].

Similar to how initial checkerboard detection algorithms used general handcrafted techniques for interest point detection, we could try to use these general feature detectors with machine learning for checkerboard detection. Yet we expect approaches specifically targeted at checkerboard detection to be more accurate and successful. However, such approaches did not really exist in the literature. Therefore we have explored the possibility of performing checkerboard feature detection with a convolutional neural net (CNN) in [Donné 2016b]. We now outline our network design and compare it with the state-of-the-art in checkerboard corner detection.

4.2.2 Checkerboard detection with deep learning

In [Donné 2016b], we have proposed an approach, coined Machine Learning for Adaptive Calibration Template Detection (MATE), that utilizes a three-layer CNN for the detection of checkerboard corners. While it is originally intended for the detection of black-and-white checkerboard corners, we also show that the same network architecture is perfectly able to detect a different type of checkerboard (a hexagonal CMY pattern). First, we give a short introduction to deep learning techniques and convolutional neural networks, and then detail the various aspects involved in designing and training the network. Afterwards, we evaluate our proposed approach for checkerboard corner detection with several state-of-the-art techniques.

4.2.2.1 Introduction to deep learning

In machine learning, the goal is to avoid manually-designed elements in the approach. Nearly all approaches rely on some parameter or other: in the most simple case this might be simply a threshold parameter, but in more complex cases we manually design an entire filter bank or a complex sequence of processing steps such as for the ROCHADE technique in figure 4.9. For many image processing tasks, the steps in a procedure consist of (or can be expressed as) convolutions with relatively small kernels, e.g. Scharr or Sobel filtering or Gaussian filtering. In a convolutional neural network (CNN), rather than manually designing or deciding on these filters, the optimal filter kernels are estimated from a training dataset.

The entire CNN consists of several processing steps, called layers, that can either be convolutions or non-linear building blocks. As a convolution is a linear operator, a CNN built up completely from only convolutions could only accurately express a linear function of its input. By introducing non-linearities such as Rectified Linear Units [Nair 2010], sigmoids [Han 1995] or other so-called activation layers the network is able to express much more complex functions of its input. The only restriction, as discussed in the next paragraph, is that the layer

must be semi-differentiable: at all points, the left and right differentials must be well-defined.

In order to find the optimal parameters of any given layer, we use a process called backpropagation [Bishop 2007]. Say that we have a layer represented by the function f parametrized with θ, with input $\boldsymbol{x}$ and output $\boldsymbol{y}$:

$$f(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{y}. \tag{4.1}$$

We have a goal for the network in mind, which we have quantized somehow with a cost function $\Phi(\boldsymbol{y})$ (if this is not the final layer of the network, then Φ also includes the processing by all following layers before calculating the cost). We stipulate that all layers as well as the cost criterion must be either left- or right-differentiable in all domain points, such that the partial differential $\partial_{\boldsymbol{y}}\Phi(\boldsymbol{y})$ is well-defined. In that case, we can express the derivative of the cost function to respectively the input and the parameters of f using the chain rule:

$$\begin{aligned}\frac{\partial}{\partial \boldsymbol{x}}\Phi(f(\boldsymbol{x};\boldsymbol{\theta})) &= \frac{\partial}{\partial \boldsymbol{x}}f(\boldsymbol{x};\boldsymbol{\theta})\frac{\partial}{\partial \boldsymbol{y}}\Phi(\boldsymbol{y}) \\ \frac{\partial}{\partial \theta}\Phi(f(\boldsymbol{x};\boldsymbol{\theta})) &= \frac{\partial}{\partial \boldsymbol{\theta}}f(\boldsymbol{x};\boldsymbol{\theta})\frac{\partial}{\partial \boldsymbol{y}}\Phi(\boldsymbol{y}),\end{aligned} \tag{4.2}$$

where $\boldsymbol{y} = f(\boldsymbol{x};\boldsymbol{\theta})$. As a CNN is built from a sequence of layers, the input of one layer is the output of its preceding layer and we can simply use the input differential $\partial_{\boldsymbol{x}}\Phi(f(\boldsymbol{x};\boldsymbol{\theta}))$ as the output differential of the preceding layer: we start at the last layer and make our way to the first one, hence the name *backpropagation*.

Optimizing the various layers is done over datasets with known ground truth outputs, the collection or generation of which may or may not be straightforward. Because of this difficulty, datasets are not infinitely large and a meaningful subdivision must be made. It is very possible for a network to overfit to a given dataset. In that case, it expresses a function that has the correct output for all examples in the training set while not inferring the correct result for unseen examples: the network is said not to generalize and to have overfit. In order to detect this, the dataset is split in two parts: the training set that is actually used to estimate optimal kernels, and a testing set that is used to evaluate the network's performance on unseen examples.

In most cases, the network still contains some adjustable parameters, so-called hyperparameters, like the spatial support of the various convolutions, or the number of convolutional layers. The optimization of these, often manually, should not be done over the training set, as there is already a certain degree of overfitting to the training set. Instead, the testing set is split into a testing set and a validation set: the results on the testing set are used to tweak the hyperparameters, while the results on the validation set indicate the overfitting behavior of the entire system. A typical split in literature is 70% training examples, 20% testing examples and 10% validation examples, if required.

This section has contained only a quick primer on convolutional neural networks to serve as an introduction and to offer the necessary background for interpretation of the following sections. The research domain is obviously much bigger than this. We refer the interested readers to recent survey papers specifically on convolutional neural networks [LeCun 2015, Schmidhuber 2015] or a more general overview of machine learning techniques for pattern recognition in [Bishop 2007].

4.2.2.2 Network design

Our proposed network architecture contains three convolutional layers, each followed by a ReLU as their non-linear activation unit, that is expected to output a single-channel response map as illustrated in figure 4.11. The first layer is intended to extract a series of features from pixel neighborhoods, while the final two layers combine these features into a meaningful chessboard corner score. The first layer of the network consists of a relatively large kernel size convolutional filter with many output channels. It is given a large spatial radius because of the good results obtained by circular boundaries [Bennett 2014, Bok 2016]. The radius of the spatial filter at this stage should be large enough to overcome the effects of the image blur on the corners, as illustrated in figure 4.10. This blur is typically the result of a badly configured focal distance at acquisition time. We have chosen a radius of six pixels (resulting in filter size 13×13), which is more than sufficient for the scenarios we evaluate—it is slightly larger than the radius used in [Bennett 2014]. Larger spatial radii would allow for more focal blur: in most cases, a larger radius is unnecessary while slowing down the network; we explore this trade-off in the results later. We have assumed that the input images are gray-scale, i.e. that there is only a single input channel. The network can also be used on color inputs; in this case, the number of hidden channels does not change and the only change is that the first layer's convolution now spans three inputs channels rather than a single one.

This proposed neural net can be interpreted as a generalized formulation of existing solutions. For example, the handcrafted approach from [Bennett 2014] can be directly mapped onto the proposed network. Let the first layer implement the various terms in the so-called sum- and difference-responses from that work, as well as the neighbor and local mean. The second layer then combines these characteristics into the sum-, difference- and mean-responses as defined in [Bennett 2014]. The final layer combines the various responses into a single response map for decision making. With similar reasonings, the methods of [Arca 2005, Zhu 2009, Ha 2009, Bok 2016] can be implemented by the proposed network: as such, the preexisting methods can be interpreted as possible solutions to the network training optimization problem. As we, through the training procedure, estimate the optimal decision rules as far as the training set is con-

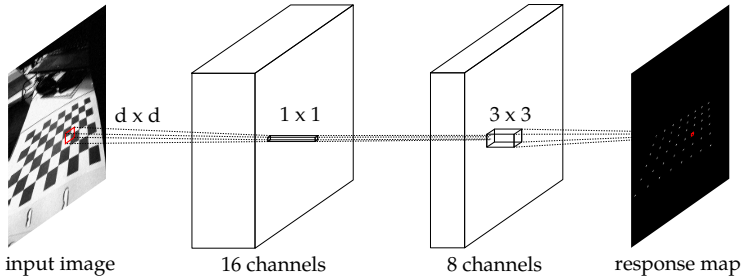


Figure 4.11: Overview of the proposed CNN for MATE. The input image is first filtered with a relatively large kernel size (proportional to image content) into 16 feature channels. The middle layer filters this locally (i.e. without any spatial support, a fully connected 16×8 *layer for each pixel with weights shared between all pixels), to introduce additional non-linearity. The last step achieves a single-channel response map. The activation functions at each step are ReLUs (rectified linear units [Nair 2010]).*

cerned, we expect the proposed method at the very least to match the performance of these existing methods. This is, of course, assuming that our training procedure is chosen well and the training dataset is sufficiently general.

4.2.2.3 The training dataset

The dataset we use for training our proposed CNN consists of two parts: calibration images as we have captured them (using a Logitech C930e Pro, at full HD resolution), and digitally distorted versions of these images (to extend the training set). The set of 85 images covers a wide range of board orientations as illustrated in figure 4.12 (they were resized to 960×540, half their original resolution). The background of these images was intentionally kept cluttered, including plants, computer screens and notable, a person holding the board – as in a realistic calibration scenario. In order to increase the size of our training set, we also include the right-angle rotated versions of these images, half of which have their intensity inverted. We call this P_{clean}, the undistorted training set.

Because the distortions and noise in P_{clean} are low, we artificially add both in a bid to make the resulting network more robust. We add both radial and tangential distortion [Heikkila 1997, Kannala 2006] as well as Gaussian noise to the clean training set to simulate poor image quality. The distortion parameters are uniformly distributed between -0.1 and 0.1, which ranges from nearly none to unlikely severe lens distortion. The added Gaussian noise has a standard deviation of 10% the intensity range, which is also relatively high. The resulting noisy and distorted training set P_{full} is illustrated in figure 4.13.

Ground truth corner locations are obtained through manual annotations: the four outer corners of the checkerboard are indicated manually on the clean im-

(**a**) (**b**)

*Figure 4.12: Two examples from the captured training data set, which ranges from clean front-view images (**a**) to oblique vantage points (**b**).*

(**a**) (**b**)

Figure 4.13: Two examples from the full training data set. These are the corresponding images from figure 4.12 with added distortion and noise.

ages, and the interior corners are interpolated. Finally, all of these points are converged locally to the saddle points in their vicinity – the resulting corners were checked manually for errors but none such were found. The ground truth locations for P_{full} were generated by applying the known distortion transformations on the corresponding locations from P_{clean}.

4.2.2.4 Training the MATE network

Our network is trained by the use of stochastic gradient descent [Bottou 2010] (SGD) for optimizing the filter weights. Rather than feeding the entire dataset through the network for each parameter update, SGD uses a subset of the entire training set – a minibatch – for each update step. If this minibatch is sufficiently large, its distribution is expected to mimic that of the entire dataset so that the parameter update will be similar to that which would have been calculated from the entire dataset. However, processing only a fraction of the entire training set results in a significant speed-up. In addition, a momentum term is used to mitigate the effect of local optima in the parameter space [Ngiam 2011, Sutskever 2013], which is often an issue when training deep neural networks.

The prediction of our network is a real number for each input pixel (based

on its neighborhood). We stipulate that a corner point should have a response of 1 or higher, while a non-corner point should have a response of 0 or lower. For the penalization function $\Phi(x_{\text{estimated}}, x_{\text{gt}})$, we have used the one-sided quadratic difference as illustrated in figure 4.14:

$$\Phi(x_{\text{estimated}}, x_{\text{gt}}) = \begin{cases} x_{\text{estimated}}^2 & \text{if } x_{\text{gt}} = 0 \text{ and } x_{\text{estimated}} > 0 \\ (1 - x_{\text{estimated}})^2 & \text{if } x_{\text{gt}} = 1 \text{ and } x_{\text{estimated}} < 1 \\ 0 & \text{elsewhere} \end{cases} \tag{4.3}$$

We penalize corner points that get a response score lower than 1, and non-corner pixels with a response score higher than 0. The goal is to enforce something similar to the principle of a maximal margin in support vector machine training: the network should in general avoid responses between 0 and 1 because those can never have zero loss. At testing time, we choose the binarization threshold to be 0.5, but this can be adjusted in either direction for more precision or more recall: whichever is preferred by subsequent processing steps.

We ignore any locations near the borders of the image during training. We also ignore pixels right next to the ground truth corner locations (within a distance of 2 pixels). We do this to make the training phase more robust against small ground truth inaccuracies. At test-time, we will select corner candidates by non-maximum suppression of the response map, and afterwards threshold the resulting maxima. This means that we can allow points in the immediate neighborhood of corners to have a high response as well: those locations are therefore "don't care". Implicitly, we assume that the network will give a relatively high score to these pixels around the ground truth corners, and will naturally learn to give the highest score to the actual corner location rather than the manually annotated corner position; evaluation of the trained network shows that this assumption holds. At the same time, we protect ourselves against inaccurate ground truth labels, and let the network itself infer the more precise location of the corners through sheer volume. We have found this trade-off between direct guidance and robustness to slightly inaccurate labeling to be worthwhile.

Additionally, we include weights for all of the responses. Because of the disparity between the number of ground truth positives and negatives, we weight the cost in each location by the occurrence of that ground truth state. This is an established method of coping with class imbalance in object classification problems, and has shown its merit for this task, too. By a different weighting between both classes, we could favor precision over recall or vice versa, but as we have mentioned two paragraphs back, we can already favor one over the other by changing the detection threshold at testing time.

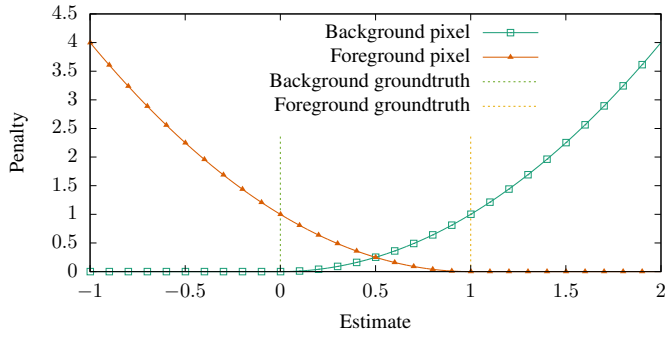
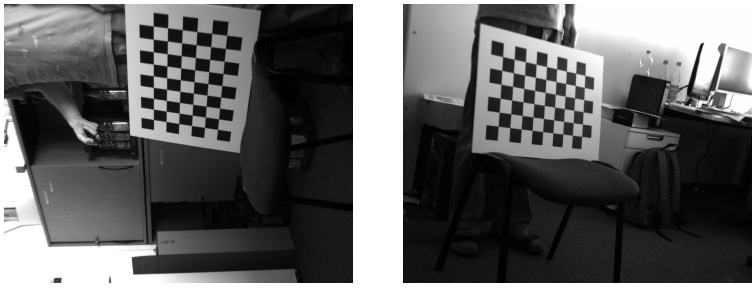
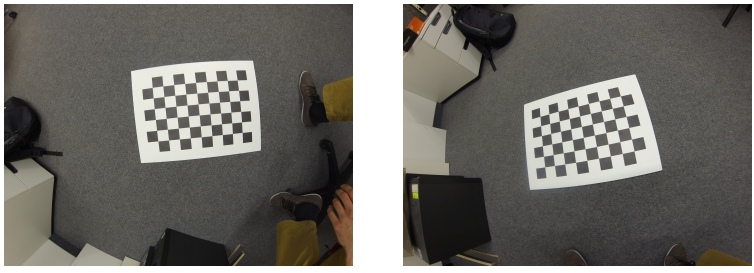


Figure 4.14: The penalty function used when training the MATE network. Pixels that should be detected as corner points are penalized quadratically when their response is smaller than 1*; other pixels have a response larger than* 0 *penalized. Enforcing such a margin is similar to the principle of a maximal margin from support vector machine training.*



(a) **(b)**

*Figure 4.15: An example from the uEye stereoscopic dataset [Placht 2014]: the left (**a**) and right (**b**) views.*



(a) **(b)**

*Figure 4.16: Two examples from the GoPro dataset [Placht 2014]: the same scene from two vantage points (**a**) and (**b**). Even for the head-on view (**a**), significant warping is present in the image.*

4.2.2.5 Evaluation datasets

We have trained two versions of MATE: the main MATE network (trained on P_{full}) and MATE* (trained only on P_{clean}). For the evaluation of these networks we use both of our created datasets, which is inherently unfair to the other techniques as our network was trained on these. Therefore, we also evaluate two datasets from Placht et al. [Placht 2014]: uEye and GoPro. In the sense of the training/testing/-validation split discussed in the introduction to deep learning, we have used a split of P_{full} or P_{clean} as the training and testing dataset, whereas the uEye and GoPro datasets serve as validation sets.

The uEye dataset is captured by two IDS UI-1241LE cameras (Imaging Development Systems, Obersulm, Germany), in a wide-baseline set-up. The lens distortion is insignificant, and the image resolution is 1280 by 1024: these images are used as-is. Some examples are shown in figure 4.15. This dataset will serve to evaluate the robustness against perspective transforms: the chessboards in the uEye dataset are typically at an angle to the imaging plane because of the wide-baseline set-up. Both MATE* and MATE are assumed to perform well on this dataset, as both are trained against perspective transforms.

The GoPro dataset, on the other hand, has a very large resolution (4000 by 3000) and a generally good quality. However, the wide-angle lens of the GoPro introduces significant lens distortion, and therefore this dataset evaluates the robustness against lens distortions. The images are used at half-resolution as the goal is simply to detect the corners and the full resolution is too much of a good thing; (sub)pixel refinement occurs afterwards at a local level anyways, and that can be done at the full resolution. All evaluated methods act on the same downsampled images. We show some examples from the GoPro dataset in figure 4.16.

4.2.2.6 Comparison of MATE to the state-of-the-art

We perform comparisons with several state-of-the-art methods for checkerboard corner detection: ChESS [Bennett 2014], ROCHADE [Placht 2014] and OCamCalib [Rufli 2008]. We evaluate the methods both on the datasets we created as well as on the uEye and GoPro dataset from [Placht 2014]. Additionally, we evaluate the methods on an "angle" dataset, which is intended to test the robustness of the methods to checkerboard skew.

Results for the training datasets are given in table 4.1 and table 4.2. The uEye and GoPro dataset results are available in table 4.3 and table 4.4, and the results from the "angle" dataset are shown in figure 4.17. In practice, we use MATE: as shown below, this method sacrifices some precision in favor of recall by allowing more false positives to make sure all corners are detected.

We evaluated the raw detections, without any sub-pixel refinement steps. Each raw detection is linked to its closest ground truth corner, and if the distance is less

Method	Accuracy (px)	Complete Checkerboards	Time (ms)
MATE*	1.009	104/104	246
MATE	1.160	103/104	264
ChESS	1.094	104/104	212
ROCHADE	1.130	104/104	6458
OCamCalib	0.758	52/52	160
Method	**Missed Corners (%)**	**Double Detections (%)**	**False Positives**
MATE*	0.000	0.000	0
MATE	0.020	2.444	40
ChESS	0.000	0.280	10
ROCHADE	0.000	0.000	0
OCamCalib	1.202	0.000	0

Table 4.1: Evaluation results for MATE on the training set P_{clean}.

than five pixels it is counted as a true positive. The accuracy gives the average distance between true positives and their ground truths. The missed corner rate and double detection rate denote how many ground truths have either zero or several detections. In practice, a sub-pixel refinement step is done after this, for more accurate corner positions, and the most interesting metric is the number of true/-false positives, and the number of missed corners, which are unaffected by such a sub-pixel refinement step. For the MATE detectors, we perform non-maximum suppression on the neural network output and then apply a threshold of 0.5. The ROCHADE method's public implementation was used; the ChESS detector was re-implemented in the Quasar language [Goossens 2018]. OCamCalib's publicly available source code was used in its evaluation. The number of detected checkerboards for OCamCalib is self-reported: this means that it may report all checkerboards as being detected even when when some corner detections are missing; this is why there are no incomplete checkerboards even with missing corners. Given the nature of the OCamCalib method, it is not meaningful to decouple point detection and checkerboard construction.

We can see from table 4.1 through table 4.4 that the trained neural nets do not lose performance over state-of-the-art techniques. While, as expected, they trump on the training datasets, they do not lose performance on external datasets (table 4.3 and table 4.4). We notice that the neural net trained on the full training set allows more false positives in return for less false negatives, a result of being trained on noisy and distorted samples. The execution time for ROCHADE remains constant because it rescales large images, while the execution time of OCamCalib varies depending on the difficulty of detecting the checkerboard. Double detections show how many corner points have more than one detection nearby.

Method	Accuracy (px)	Complete Checkerboards	Time (ms)
MATE*	0.810	66/104	242
MATE	0.999	81/104	246
ChESS	1.042	73/104	209
ROCHADE	0.423	38/104	6642
OCamCalib	1.084	52/52	243
Method	**Missed Corners (%)**	**Double Detections (%)**	**False Positives**
MATE*	1.162	0.020	0
MATE	0.62	4.147	40
ChESS	0.842	0.902	15
ROCHADE	54.899	0.000	0
OCamCalib	30.967	0.000	1

Table 4.2: Evaluation results for MATE on the training set P_{full}.

Method	Accuracy (px)	Complete Checkerboards	Time (ms)
MATE*	0.886	181/206	531
MATE	1.009	186/206	529
ChESS	0.946	175/206	473
ROCHADE	1.510	186/206	6753
OCamCalib	0.319	206/206	261
Method	**Missed Corners (%)**	**Double Detections (%)**	**False Positives**
MATE*	3.497	0.009	12
MATE	3.065	0.809	492
ChESS	3.398	0.000	11
ROCHADE	2.895	0.000	1
OCamCalib	0.000	0.000	0

Table 4.3: Evaluation results for MATE on the uEye dataset from [Placht 2014].

Method	Accuracy (px)	Complete Checkerboards	Time (ms)
MATE*	1.323	81/100	1209
MATE	0.835	86/100	1205
ChESS	1.389	80/100	1080
ROCHADE	1.807	80/100	6688
OCamCalib	0.458	100/100	533

Method	Missed Corners (%)	Double Detections (%)	False Positives
MATE*	10.556	0.000	12
MATE	4.556	4.556	389
ChESS	5.481	0.222	56
ROCHADE	5.593	0.000	3
OCamCalib	0.537	0.000	0

Table 4.4: Evaluation results for MATE on the GoPro dataset from [Placht 2014].

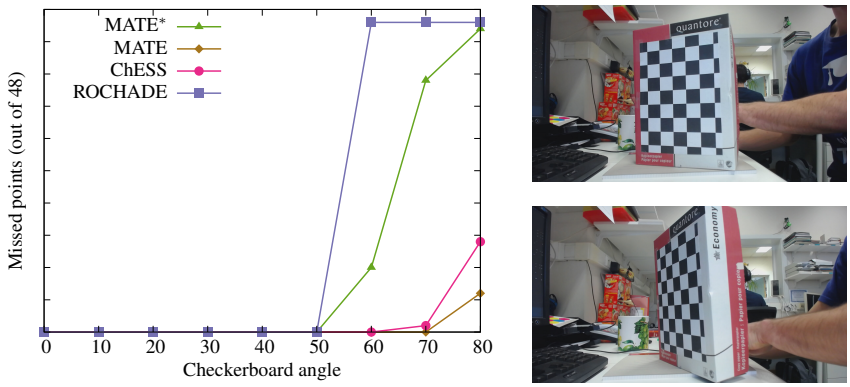


*Figure 4.17: The results from the 'angle' dataset. (**a**) the checkerboard angle against the number of missed points; (**b**) example images (20 and 60 degrees, respectively).*

Notably, MATE* does not lose much accuracy over MATE on the GoPro dataset, even though this dataset contains high degrees of lens distortion—a scenario MATE* was not explicitly trained for. We conclude that the various types of perspective transforms in MATE*'s training set conferred enough robustness to handle these distortions. The large difference in accuracy between MATE and MATE* on the full training set is the result of the noise, a factor for which MATE* was woefully ill-prepared.

In the noise-free datasets, OCamCalib performs the best out of the tested methods, including the proposed method: it detects all of the chessboards. However, it counts a checkerboard as detected even when some points are still missing – in contrast, the other methods only count a checkerboard as completely detected when all of its points were detected. The percentage of missed corners gives a much better indication in this case. There are two main additional drawbacks to OCamCalib: it requires the dimension of the checkerboard to be known in advance, and it requires the chessboards to have a white border. This is an issue on the training sets, in which half the checkerboard have a black border because of the intensity reversal. For this reason, this method was only run on the non-inverted half of those datasets. None of the other methods had any issue with the inversion of the images, and they were evaluated on the entire datasets.

We conclude that the trained neural network is able to match or outperform other, hand-designed, corner-point detectors in the tested scenarios. In noise-free settings, it is hard to beat the performance of OCamCalib, which uses higher-level info (the checkerboard size and inter-point relations) to detect the checkerboard. However, in the noisy dataset included here, OCamCalib misses the detection of a large number of points, which our proposed method as well as the ChESS method [Bennett 2014] do detect.

Figure 4.17 shows how the various methods cope with the detection of a chessboard under various angles. The neural net trained on the full training set is able to detect the full chessboard under a 70-degree angle, while the other methods lose the detection of the full chessboard earlier. Even though the training set does not explicitly include images of chessboards under such extreme angles (otherwise, the network trained on the clean set would also have a similar performance), the lens distortions simulated in the full training set mean that MATE is able to better cope with large distortions, apparently perspective ones as much as lens distortions. This feature is of particular importance in multi-camera set-ups: it is far less likely that all of the imaging planes will be parallel, and hence the chessboard will be at significant angles to the various imaging planes – which poses a significant challenge for the detection methods.

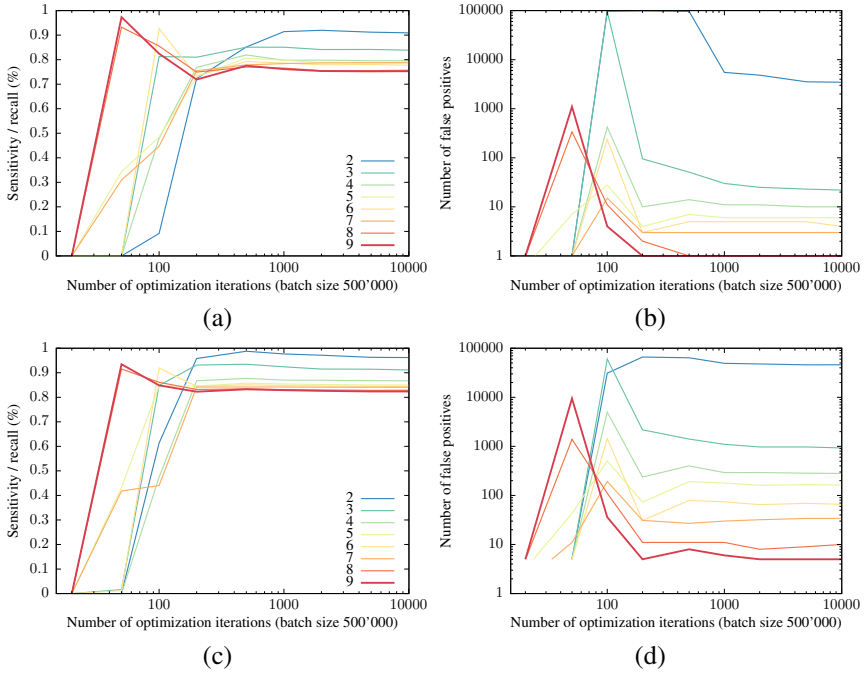


Figure 4.18: Recall of the checkerboard corners (the fraction of corners detected as such) and the number of false positives in the uEye and GoPro datasets for networks with varying spatial support radii, as a function of the training iterations. (a) and (b): recall and false positives for the uEye dataset, (c) and (d): recall and false positives for the GoPro dataset.

4.2.2.7 Impact of the spatial support for the first layer

In this subsection, we explore the parameter space for the radius of the first convolutional layer. We train the network, with a varying spatial support, on P_{full} for $10,000$ update steps with a minibatch size of $500,000$ patches. The performance of the various networks, in terms of their recall and number of false positives, is shown in figure 4.18. The required execution time for the various spatial support radii is shown in figure 4.19.

Note that the training phase optimizes the number of relative mistakes—the percentage of missed checkerboard corners versus the percentage of background pixels falsely detected as checkerboard corners. This is done by weighting the error terms of these two subsets based on their cardinality. As mentioned earlier, this alleviates the class imbalance issue. Networks with a large spatial radius are better able to suppress false positives than networks with smaller supports. Networks with small spatial supports (two and three pixels, mainly) simply do not have the requisite expressive power: up to several hundred false positives per frame for a

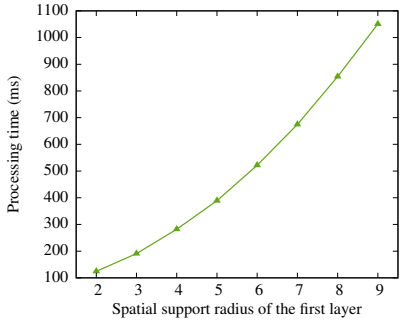


Figure 4.19: Average processing time required for the various spatial supports in the first layer, for an image in the uEye dataset. This time includes the application of the threshold, the non-maximum suppression and the creation of the list with detected pixel locations.

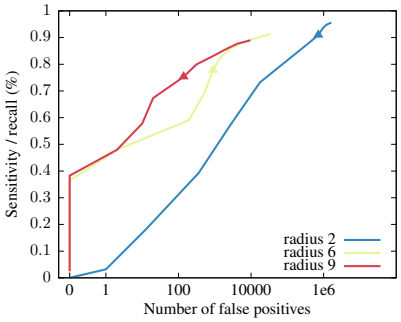


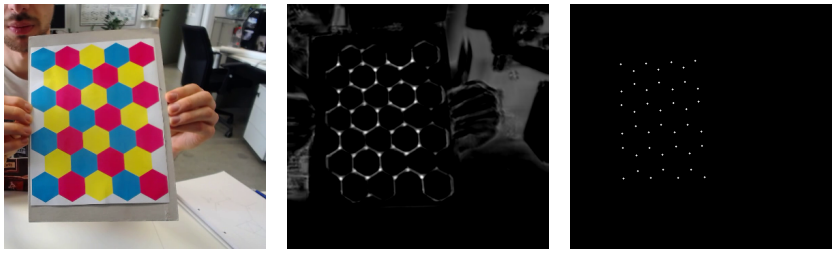
Figure 4.20: The curves for recall and number of false positives in the function of the threshold used for the response maps of the proposed neural network, with the point for a threshold of 0.5 marked. Curves closer the top left corner are better.

spatial support of two pixels on the GoPro dataset. We have chosen for a network with a spatial support radius of six pixels for the earlier comparisons, which allows us to compare more meaningfully with CheSS [Bennett 2014], which has the same spatial radius.

Additionally, we remind the reader that our proposed CNN approach already has a method for the trade-off between recall and precision: the threshold applied to the response maps output by the network. Figure 4.20 shows the recall-precision curves for networks with radii 2, 6 and 9, in function of the applied threshold. From this plot, we conclude that, while the choice of threshold can be used to trade between recall and precision, (well-trained) networks with larger spatial supports will offer better performance – a higher precision-recall curve – at the cost of higher processing time.

4.2.2.8 Detecting other patterns

We now illustrate that attuning MATE to specific scenarios is much less labor intensive than manually designing a new procedure; it only entails annotating the requisite ground truth. Say that we wish to use a CMYK calibration plane to stamp down on false positives and increase the density of measurements on the calibration object. Using a hex tiling, all corners are equivalent up to a 120 degree rotation and/or a reflection. We will not get into a discussion on the merits or disadvantages of a non-square calibration object; it merely serves as an illustration of the flexibility of MATE. After a training phase similar to the one outlined in section 4.2.2.4, the network is able to detect the corners of the hexboard well, as illustrated in fig-



(**a**) (**b**) (**c**)

*Figure 4.21: Result for the neural net trained on a CMYK hexboard input. The input image (**a**), the response map (**b**), and the detected calibration template corners (**c**).*

ure 4.21. Together with the relatively low number of training images (on the order of 100, which can be annotated manually in an acceptable timeframe) this is especially interesting for scenarios where the traditional checkerboards are unfeasible and non-standard configurations need to be used, as discussed in section 4.1.5. The time required for annotating the relatively small training dataset with ground truth is far lower than would be required to design a tailored feature detection algorithm from scratch.

4.3 Contributions

In this chapter, we have presented an overview of the most popular calibration objects, organized according to their dimension: 0D spheres, 1D ranging poles, 2D checkerboards and 3D calibration rigs. Such objects are easy to detect in natural images, and yield many correspondences between 2D locations and known 3D points to use for camera calibration. Of these, checkerboards are by far the most popular, due to the combination of being easy to construct and handle as well as providing many 2D to 3D correspondences per captured image. We have discussed the literature on detecting these 2D checkerboards in images, and motivated why a deep learning technique might hold important benefits for calibration object detection. Accordingly, we have presented our deep-learning method for checkerboard detection, MATE, which is quick, accurate, and robust to noise and distortions of the images. We illustrate that it is straightforward to attune to new scenarios, forgoing the necessity for manually crafting feature descriptors and detection algorithms. The network design of MATE and the accompanying discussion was published as an article in a peer-reviewed journal:

- *MATE: Machine Learning for Adaptive Calibration Template Detection*, Simon Donné, Jonas De Vylder, Bart Goossens and Wilfried Philips. MDPI Sensors, 2016 [Donné 2016b].

Towards the future, we are interested in leveraging the power of deep learning for camera calibration in more complex ways. With MATE, we succeed in robustly detecting the checkerboard corners, but after that we still require the link between the observations and specific checkerboard corners. The advantage of MATE is that it has a low spatial support, and is therefore a fast approach. By using much larger spatial supports, we could hope to have the network also infer these correspondences. Even more interestingly, we could design a new pattern (visualize, for example, a grid of QR codes) for which the correspondence becomes possible based solely on the local neighborhood of the corner.

5

State-of-the-art of 2D-to-3D reconstruction

" So yes, in some ways, these would be enough. But how awful would that be? How terrible, to live surrounded by the stark, shar, hollowness of things that were simply enough? "

— Patrick Rothfuss, *The slow regard of silent things.*

As the title of this work implies, our end goal is to estimate the 3D structure of an observed scene. Having obtained a model of the cameras in the scene by estimating their parameters in chapter 3, we know accurately how 3D locations are projected onto the 2D images; now, we we wish to invert this projection. In this chapter, we give an overview of the many ways one can go about estimating the 3D model of the scene from a set of 2D images or point observations: 2D-to-3D reconstruction. We consider two different aspects for categorizing 2D-to-3D reconstruction techniques: what type of inputs/outputs the method consume and produce, and secondarily whether or not they actively influence the scene.

The input observations are either single point observations (with or without known correspondences between the observations by different cameras), they are entire RGB images, or even already depth maps that need to be combined into a single 3D model. The methods may estimate a point cloud in 3D, or even just one single point, they may estimate depth maps for some or all input views, or they may reconstruct meshes or other volumetric representations. Active methods interact with the scene and measure the reaction in order to extract 3D information:

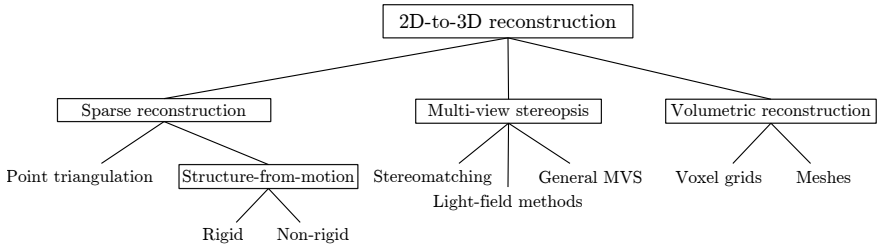


Figure 5.1: An overview of the taxonomy of the 2D-to-3D reconstruction techniques and the organization we use in this chapter.

laser rangefinders, for example, project a laser pulse into the scene and measure the round-trip time to estimate the distance to a single point. On the other hand, passive methods do not actively influence the scene, and rather estimate 3D information based only on whatever information the scene itself emits. The taxonomy we use in this chapter is shown in figure 5.1, where we have only passive methods, which we focus on in this work.

The most fundamental problem is that of reconstructing a single point. Assuming that they can be controlled and pointed in the right direction, laser rangefinders are good active sensors for estimating the depth of separate points. However, due to the high speed of light, an extremely precise measurement of the round trip time is required and high precision sub-millimeter accuracy is unfeasible in general; laser rangefinders are mostly useful for long distances. Instead, single-point triangulation offers an alternative by estimating the point location from measurements by several calibrated cameras. As a passive method, triangulation does not require us to actively interfere with the scene.

While point triangulation is useful for reconstructing many separate points, we are often also interested in modeling coherent point clouds. As many objects are static or at least strongly restricted in the freedom of their deformations, jointly reconstructing the points constituting a given object allows us to place stringent assumptions on the behavior of the underlying point cloud. Rigid Structure-from-Motion (RSFM) and Non-Rigid Structure-from-Motion (NRSFM) techniques leverage this in order to drastically reduce the number of parameters that need to be estimated.

Yet it is often not enough to have information about only a sparse set of scene points, and we require depth information about all observed points. Several active measurement devices exist for this scenario. Time-of-flight (ToF) cameras and Laser Imaging Detection and Ranging (LIDAR) devices are in essence built from a large number of laser rangefinders, either in a traditional camera housing in the case of ToF cameras or, in the case of LIDAR, on a rotating head adopted from radar and sonar devices. The drawback here is again the relatively low ac-

Figure 5.2: Illustration of the 2.5D nature of depth map reconstruction for the input image in the top left from [Scharstein 2007]. While the front view (left) of the reconstruction looks okay, the missing parts become more than obvious in the other views.

curacy for small-scale scenes, as well as the often prohibitively high cost, both for the purchase and for the power requirements in the case of a mobile system. An alternative active method is given by structured-light scanning. By projecting a narrow band of light onto the surface of 3D object, and observing it from other perspectives than that of the projector, the distortion of the line can be used for an exact geometric reconstruction of the surface shape. While multiple lines can be projected simultaneously for a quicker capture process, using too many of them results in interference, and the capture process is in general somewhat slow, but more importantly its application is restricted to controlled environments. There is also the possibility for passive dense depth estimation: multi-view stereopsis techniques use the correspondences between multiple images to perform point triangulation at an extremely large scale. The challenging part of the process is the estimation of the correspondences: we need to know which pixel in one image corresponds with a given pixel in the other, in order to perform this triangulation. A more restricted version of the multi-view stereopsis problem simulates the function of the human visual system: when two cameras are placed on a baseline and look orthogonal to that line, the resulting depth estimation problem boils down to find a given pixel's correspondence in the other image – with the guarantee that the correspondence lies on the same image row as the original pixel, which simplifies the algorithms.

Unfortunately, depth map estimation results only in a partial reconstruction of the scene. Yes, we may have (often successfully) estimated the depth for all observed pixels in the scene. However, the scene reconstruction from a single depth map is said to be only 2.5D rather than 3D: we illustrate this in figure 5.2.

If we wish to acquire a more complete reconstruction of the scene, we require a way to either merge all these depth maps, or to forgo the depth maps completely and estimate the 3D reconstruction in another way. We can represent the scene in full 3D using meshes, point clouds or a volumetric representation based on a

quantized sampling grid, a so-called voxel grid. It is typically easier to reason on these full 3D representations, e.g. for tasks like path finding or object recognition. Meshes and point clouds are relatively efficient in terms of memory usage, but require more involved methods, e.g., for calculating occupied space or possible paths through them. A volumetric representation sports more straightforward scene interpretation and analysis, at the cost of high memory usage – because the number of voxels increases cubically with the voxel resolution and with the scene size, the resolution of the voxel grid is limited in practice. In comparison, representing the scene with a set of depth maps makes subsequent reasoning and tasks more difficult and typically requires a conversion to one of the other representations.

Finally, we also discuss the Simultaneous Localization and Mapping (SLAM) literature. They share many of the problems and techniques above, as they simultaneously estimate both the camera positions and the scene structure. Their main drawback is that, while they can often handle slightly deforming scenes by filtering out moving objects and removing them from the scene model (e.g. other cars moving in a street scene for an autonomous vehicle), strongly deforming scenes are not handled well. They were not included in figure 5.1, because they can fall into any of the categories there depending on the specific variant.

We now expand on the various techniques to present a clear idea of the drawbacks and advantages of each. We discuss our improvements to several of these methods in later chapters, and postpone relevant technical details until then.

5.1 Sparse reconstruction

With sparse reconstruction, we denote the location estimation of a small number of distinct points in the scene, collecting the triangulation and SFM/NRSFM under this name. Based on 2D point observations from several different vantage points, we estimate the locations of the observed 3D points. In this discussion we assume that point correspondences are known, but this is a non-trivial assumption: it means that we know perfectly well which observations in the image set correspond to the same physical point, without any errors. In practice, however, mistakes will be present in the correspondences. Such outliers can be detected and discarded based on their effect on the cost function [Olsson 2010], but discussing the methods for estimating these correspondences, while interesting, would take us too far. Instead we focus on improving the estimation itself, the result of which is useful for outlier detection: in that way, improvements in the sparse reconstruction process propagate to outlier detection algorithms.

We first discuss the localization of a single feature point based on a set of observations. It is straightforward to see that, in the case of precalibrated cameras, the reconstruction of a point cloud with known correspondences separates into a large number of single-point reconstruction problems.

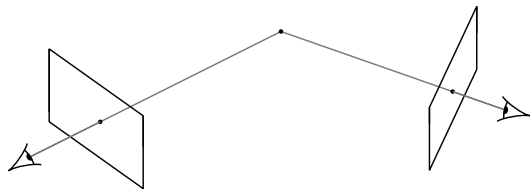


Figure 5.3: Single-point triangulation can be done by tracing back the rays of the observations into the scene: the 3D point that caused these observations should be close to all of the rays from the various cameras. Without any noise of any kind (in the observations or in the camera parameters), these rays intersect in a single point, as shown above. In practice, of course, they do not, and we wish to find the best estimate for the underlying 3D location.

In a more general version of the problem, the positions of the cameras aren't known perfectly any more, although we still assume that the intrinsic parameters are well known; as these parameters typically remain constant, this is not a very difficult assumption to fulfill. Under the assumption that the observed objects are rigid, and that the point correspondences are still known, rigid structure-from-motion techniques jointly extract both the camera positions and the point cloud structure. Yet many objects are not rigid; even though their freedom is severely constrained due to physical considerations, they still deform noticeably. Freely deforming objects are hard to reconstruct as we have less measurements than there are unknowns; non-rigid structure-from-motion techniques impose constraints on the scenes deformations to lower the number of unknown parameters to the point that problem becomes well-posed.

We now discuss the literature for point triangulation and rigid and non-rigid structure-from-motion each in their own sections.

5.1.1 Point triangulation

Triangulation is the problem of estimating the 3D location of a physical point based on observations by multiple well-known cameras. As shown in figure 5.3, we can trace back the rays of the observations into the scene. The point that caused these observations should lie close to all of the rays by the various cameras; in the ideal noiseless case, all rays intersect in a single point. In people tracking methods, for example, these feature points can be the centroids of the foreground blobs in a foreground/background segmented video sequence.

In more complex problems, these correspondences between various views can be computed using a variety of feature descriptors such as SURF [Nga 2013] or ORB [Rublee 2011]. This can be done by first selecting a set of candidate observations and then performing the triangulation on this group of observations. If the resulting triangulation error γ is too large, then the selected features were likely not all observations of the same point. Olsson et al. show that it is not necessary

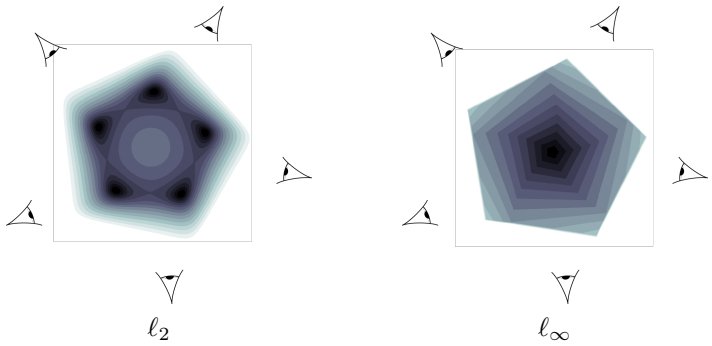


Figure 5.4: A visualization of the values of the cost function in point triangulation (darker colors are better solutions, according to the corresponding cost function). The actual point location is in the center. For a well-chosen triangulation problem with five cameras, the multiple local minima for ℓ_2-norm aggregation are obvious (and the actual location, in the center, is not even a local minimum), while the cost field for the ℓ_∞-norm aggregation does not contain local minima. This is a more general version of the example by Kahl and Hartley [Kahl 2008b], who showed the same for a scene with only three cameras.

to finish the triangulation procedure in order to detect such outliers [Olsson 2010]. Yet the performance of such methods for correspondence tests clearly depends crucially on the speed of solving the triangulation problem.

When performing the triangulation, we have a choice for which error norm to minimize. The accuracy of a location hypothesis is quantified by the difference between the hypothetical point observations and the actual measurements. Traditionally, these differences are penalized with the ℓ_2-norm, in agreement with a Gaussian noise model. However, due to the non-linearity of the camera projection the cost function can have multiple local minima when defined in terms of the ℓ_2 norm [Hartley 1997, Byröd 2007, Hartley 2013], as illustrated in figure 5.4. Early methods [Hartley 1997, Stewenius 2005] can get stuck in local minima caused by the non-linearity of the camera projection. Resolving this issue can be done through a complex, and costly, branch-and-bound approach [Kahl 2008a] or by solving for the entire set of stationary points [Byröd 2007].

The problem of local minima can also be resolved by using the ℓ_∞-norm to penalize the measured differences. We have two elements in the penalization: first of all, we need to quantify the error within a single view (the reprojection error), but secondly we also need to aggregate the errors from all views. Olsson et al. [Olsson 2007] show that the resulting cost function when using the ℓ_∞-norm for view aggregation is a *pseudo-convex* function. This pseudo-convexity implies that the set of stationary points is convex: no local optima exist and suboptimal local minima are therefore not an issue [Olsson 2007, Agarwal 2008, Kahl 2008b, Olsson 2010, Dai 2012, Eriksson 2014].

Yet many existing methods use a mixed norm: the ℓ_∞-norm for aggregation over the views in order to avoid local optima, but the ℓ_2-norm for the reprojection error in a single camera [Olsson 2007, Agarwal 2008, Kahl 2008b, Dai 2012]. Minimizing this mixed-norm aggregated error leads to a series of auxiliary second-order cone problems which are complex and time-consuming to solve. In the work of Kahl and Hartley [Kahl 2008b] the bisection algorithm was introduced for optimizing the hybrid cost function, which performs a binary search for the minimal error. Olsson et al. [Olsson 2007] also propose an approach based on a series of auxiliary problems, but not based on bisection: for their method, the auxiliary problems are local approximations to the original problem such that the Karush-Kuhn-Tucker criteria are a simple approximation to the Karush-Kuhn-Tucker criteria of the original problem. In [Agarwal 2008], Agarwal et al. present a survey of the ℓ_∞-norm for aggregating reprojection errors, as well as the Gugat method which outperforms the techniques of Olsson et al. [Olsson 2007] and Kahl [Kahl 2008b] et al. while still being based on a series of Secondary Order Cone Problems (SOCPs). Finally, the authors of [Dai 2012] show how the solution to the previous SOCP problem can be used as initialization to the next iteration's problem in order to speed up the solver.

In contrast to these hybrid methods, full ℓ_∞ methods also express the single view reprojection errors in terms of the ℓ_∞-norm. In [Eriksson 2014], the authors propose a split-Bregman approach to the optimization problem which results in an elegant modification of existing bundle adjustment (BA) packages. In each iteration of the algorithm, a single bundle adjustment iteration is performed, as well as one evaluation of a proximal operator (which requires finding the root of a one-dimensional function). The authors show that it is faster than the Gugat method from [Agarwal 2008], but it still requires the use of existing software packages such as SBA [Lourakis 2009]. In chapter 6, we discuss the geometrical method we have developed, which is more efficient in its execution and is additionally much easier to implement, without requiring the support of existing solvers or software packages.

5.1.2 Rigid Structure-from-Motion

In the previous section we have discussed how to perform the triangulation of a single point, and for this we assumed that the cameras in the scene were well known. Under certain circumstances, this is not a valid assumption: while intrinsic calibration of the cameras is often possible inside the lab using the discussed calibration objects, we can not always introduce such calibration objects to the scene. We now look at alternative methods that assume the cameras to have been intrinsically calibrated, but use the scene itself and its objects to extrinsically calibrate the cameras at reconstruction-time.

In Rigid Structure-from-Motion (RSFM), we assume that we have a set of 3D points for which we know the correspondences. RSFM gained early popularity due to its low computational complexity, but initially assumed an orthographic camera model [Webb 1982]. When the scene is relatively far from the camera compared to its size, the depth of the scene points is approximately constant over the scene, and hence dividing by the depth in the perspective camera model is roughly the same as dividing by a camera-dependent scaling factor s_c in the orthographic camera model (see section 2.1.4). For many scenes this is therefore a reasonable, and valuable, approximation. More importantly, under this camera model, the projection relationship becomes linear and the entire set of projections is expressed as a single matrix equation:

$$\underbrace{\begin{bmatrix} \ddots & \vdots & \iddots \\ \cdots & \boldsymbol{u}_{c,n} & \cdots \\ \iddots & \vdots & \ddots \end{bmatrix}}_{2C\times N} = \underbrace{\begin{bmatrix} & \vdots & \\ [s_c R_{xy,c} & & s_c \boldsymbol{t}_{xy,c}] \\ & \vdots & \end{bmatrix}}_{2C\times 4} \underbrace{\begin{bmatrix} \cdots & \begin{bmatrix} \boldsymbol{x}_n \\ 1 \end{bmatrix} & \cdots \end{bmatrix}}_{4\times N}, \tag{5.1}$$

where $\boldsymbol{u}_{c,n}$ is the observation of the n'th point $\boldsymbol{x}_n$ by the c'th camera described by the first two rows of its projection matrix: $\begin{bmatrix} s_c R_{xy,c} & s_c \boldsymbol{t}_{xy,c} \end{bmatrix}$. The problem now devolves into a simple factorization problem, which is a well researched problem in linear algebra [Webb 1982]. This seminal paper by Webb et al. has seen few direct improvements; because of its good performance,further research has focused on how to handle non-rigid scenes [Tomasi 1992, Bregler 2000, Paladini 2010, Gotardo 2011b, Dai 2014].

5.1.3 Non-Rigid Structure-from-Motion

We now relax the problem statement of RSFM by allowing the object to deform in between the different camera views, for example because the camera is moving and the different views are actually subsequent frames over time. Whereas we could previously assume that the location of point n was independent of the image index c, this is no longer the case, and we now need to estimate the locations $\boldsymbol{x}_{n,c}$ for each of the points in each of the images. This means that we wish to estimate $3NC + DC$ parameters (where D is the number of parameters in the camera matrix) using $2NC$ observations — a horribly ill-posed problem without exploiting additional prior knowledge. A wide range of methods in literature address this problem. We classify them here based on the prior they use: template-based, shape basis methods, and trajectory basis methods.

The first option is to assume that the object being reconstructed is well-known and that all of its possible behaviors have been collected into a template. As illustrated in the existing literature, a linear space often adequately represents deformations of real physical objects [Blanz 1999, Wang 2008, Paysan 2009]. Such a

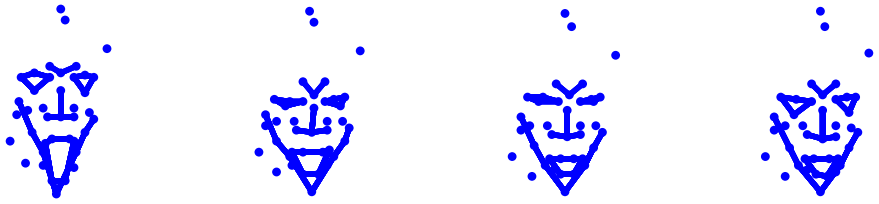


Figure 5.5: Example of a shape basis extracted by a shape basis method [Bregler 2000]. Without any knowledge about the type of scene, such methods often still extract an intuitive shape basis such as containing the collection of extreme expressions as shown here.

K-dimensional linear template effectively restricts the possible 3D point clouds to a K-dimensional linear subspace of the $3N$-dimensional space. Now we must only estimate $KC + DC$ parameters per input image. The obvious drawback is that an accurate and complete template must be available a priori – any deformations not covered by the template will not be reconstructed well.

It is exactly this lack of accurate knowledge about the deforming object which is often what prohibits the use of template-based methods. Shape basis methods, on the other hand, start from the same assumption that the object is indeed restricted to a K-dimensional linear subspace, but that this subspace is unknown and must also be estimated. Now, we need to estimate $KC + DC + 3NK$ parameters based on $2NC$ inputs [Bregler 2000]. Figure 5.5 shows an example of an estimated shape basis for a simple face point cloud, containing several natural deformations of the face. In the seminal work by Bregler et al. [Bregler 2000], the shape basis and the coefficients for each frame are extracted through matrix factorization. Rather than being restricted to a linear manifold for the subspace of possible point clouds, several works have evaluated the use of non-linear subspaces [Rabaud 2008, Shaji 2008, Fayad 2009, Gotardo 2011b]. As we are focusing on human face modeling, these non-linear models are not required; a linear shape basis is sufficient.

While Bregler et al. originally assumed an orthographic camera model, later work extended their work to a perspective model [Jing Xiao 2005, Hartley 2008]. The specific application we have evaluated is that of human face reconstruction; in this case the range of depths is typically small compared to the distance to the camera, so that an orthographic camera model is an acceptable approximation.

The final group of methods we distinguish are the trajectory-basis methods. Whereas shape basis approaches attempt to model spatial coherence, trajectory basis methods exploit temporal coherence: there is a high correlation between subsequent locations of a given point. These methods model the point trajectories over the sequence as elements of a K-dimensional subspace of their $3C$-dimensional space, rather than restricting the $3N$-dimensional point cloud per frame. The optimal basis for each input sequence can be estimated through principal component

analysis just as for the shape basis methods, but research has shown that the extracted bases are very close to the basis of the Discrete Cosine Transform so that that basis can be used in practice, forgoing the constant estimation of a trajectory basis [Akhter 2009, Akhter 2011]. A related idea is to model the camera as smoothly moving [Gotardo 2011b] too; based on the resulting camera estimates, it is possible to reconstruct the scene more effectively.

For both the shape basis and the trajectory basis methods, the dimensionality of the subspace is an important consideration. If this dimensionality is too low, the model is not expressive enough; a too high dimensionality causes overfitting as the model tries to explain noise as meaningful deformations. In chapter 7 we discuss the 2D-to-3D reconstruction of human faces, representing them with 68 feature points. In this case, a shape basis of dimensionality 8 to 12 was found sufficient. Such a base is able to model the deformations of a talking face well, displaying recognizable emotions, without modeling input noise or overfitting on the specific input sequence.

An important aspect of NRSFM methods is the issue of input noise or missing points. By explicitly modeling the noise impact and using maximum-likelihood estimators or maximum-a-posteriori estimators, Bayesian techniques that model extreme distortions as being rare drastically improve the performance of NRSFM techniques [Torresani 2008, Zhou 2012]. As far as missing points go, we will assume they are only caused by self-occlusion: part of the face being invisible to the camera because the face is turned away. Lee et al. have showed that it is in this case well possible to regress the missing point observations from the ones that are observed [Lee 2011]. This step acts only on the 2D observations, and can hence sanitize and complete the input to any of the above non-rigid structure-from-motion methods, and we will therefore assume that there are no missing points.

Unfortunately, all of the above methods focus on the off-line processing of an entire video sequence. Early research work exists that attempts to map these methods onto on-line methods, with varying success. Paladini et al. initialize the shape basis model from a short bootstrap sequence. They expand their shape basis model whenever they encounter a deformation that is not adequately handled by the current model, using principal component analysis [Paladini 2010]. While it performs well in noise-free scenarios, the presence of noise stops it from working well; it accumulates noisy shape bases as it tries to explain the noisy measurements, resulting in an ever-growing shape basis that overfits the input data. An alternative is the use of Incremental Principal Component Analysis [Tao 2013]. Using a frame window to step through the input sequence, the shape basis is constantly updated using Incremental PCA. However, this causes issues when the deformations in the scene are erratic: some modes are not seen for a long time, causing them to be forgotten and removed from the shape basis. In chapter 7, we propose an on-line variant of the method by Torresani et al. [Torresani 2008]. Due to the Bayesian

Figure 5.6: Free-view stereoscopic imagery. It is possible to trick your brain into perceiving 3D by feeding the right input images to your eyes. To do this, cross your eyes until the left and the right image overlap, and then try to focus on the single resulting image. For very complex scenes, such as the bottom one from the Middlebury 2014 dataset [Scharstein 2014], this lets a human interpret the scene a lot more easily; the same goes for computers – depth information is invaluable. Keep an eye out throughout this chapter for similar image pairs, which you can inspect in the same way.

formulation that explicitly takes noise into account, it provides a probabilistically meaningful shape basis representation. We continuously extract the shape basis from a carefully curated keyframe representation of the past observations, and so both avoid the indefinite growth of the shape basis as well as allow the long-term capture of rare deformations.

5.2 Multi-view Stereopsis

Multi-view Stereopsis (MVS) is the process of reconstructing the 3D scene based on images captured by several different cameras: either one camera that moves through space and creates multiple images of a static scene, or separate physical cameras taking a picture at the same time. When viewing the scenes from different vantage points, the so-called parallax effect states that the behavior of the projection of a given physical point in the images is influenced by its depth from the cameras. Vice versa, we can estimate depth information based on point correspondences between the images. As we have discussed in chapter 2, perspective cameras project points from camera coordinates to the image plane coordinates by dividing by the Z coordinate. Consequently, an offset along the 3D X or Y axes will have a different impact on the projected coordinates based on the point distance z; points further away have their projection change less.

In MVS, we estimate depth maps for the input images, which are afterwards optionally fused into a volumetric 3D representation. We discuss three groups: stereo matching, lightfield methods, and general multi-view stereopsis methods. Stereo matching methods aim to replicate the human physiology by using two side-by-side cameras that serve as eyes. We outline below that this set-up results in a simplification of the underlying math, allowing straightforward depth cal-

culations from the correspondences, and makes the correspondence search more straightforward. Light-field methods expand on this set-up by allowing more than two cameras on a single baseline: issues with occlusions in stereo matching are strongly mitigated in multi-view approaches. General multi-view stereopsis approaches make no assumptions about the camera set-up at all, but as a result do not have access to the same priors as do stereo matching and light-field methods.

5.2.1 Stereo matching

In human physiology, the eyes serve as the two vantage points. When alternating between closing the left eye and closing the right eye, objects in our field of view shift horizontally, and things closer to us have a higher disparity than objects further away: this is the inverse distance effect affecting the projection, the previously mentioned parallax effect. The inverse can be used to trick our brain into viewing 3D from a flat page: by carefully creating images for the left and right eye, and viewing them as such, we see a 3D scene where there is none. Several techniques for this exist, such as the iconic blue/red goggles or the more complex polarized goggles from contemporary 3D cinemas. There is also a technique that does not involve any additional hardware; so-called free-viewing of stereo pairs. Figure 5.6 shows two example pairs and some explanation as to how to view them. Some images in this work are created as free-viewing stereo pairs, and list this in their description. However, not everyone easily gets the hang of free-viewing stereo pairs. Rest assured: all images contained in this chapter are also interpretable in their flat 2D representation.

We first discuss the geometry of stereo matching, and the principle behind the so-called rectification of the input images, before discussing the current literature.

5.2.1.1 The geometry of stereo matching

Consider two cameras, C_L on the left and C_R on the right, observing a 3D point $\boldsymbol{x}$ as illustrated in figure 5.7. As mentioned before, we assume that the intrinsic and extrinsic parameters for these cameras are well known. We call the line segment connecting both camera centers the baseline; this name is also used for the length of this line segment. Typically, there are no relevant world objects between both cameras as the baseline is either the result of camera movement, or occupied by the physical construction connecting the cameras. Then for each point $\boldsymbol{x}$, the baseline and $\boldsymbol{x}$ uniquely define a plane: the epipolar plane. An interesting implication is that the projections of $\boldsymbol{x}$ on the images, respectively $\boldsymbol{u}_L$ and $\boldsymbol{u}_R$ lie on the intersection of the epipolar plane and their respective image planes.

As illustrated in figure 5.7, this plane can equivalently be defined by the baseline and either of the projections. So for a measured projection in one image, either $\tilde{\boldsymbol{u}}_L$ or $\tilde{\boldsymbol{u}}_R$, we can construct the *epipolar line* in the other image: the intersection

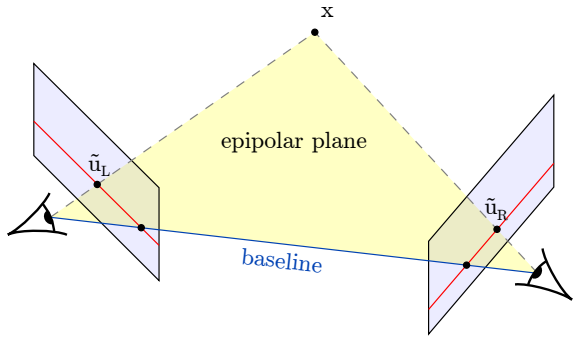


Figure 5.7: An overview of epipolar geometry. Given two cameras, we call one of them the left camera and the other the right camera. When observing a point $\boldsymbol{x}$, this point and both camera centers define the epipolar plane (yellow). The projection of $\boldsymbol{x}$ on either camera's image plane lies on the intersection of the epipolar plane and that image plane. As the epipolar plane can equivalently be defined by both camera centers and either of the projections of $\boldsymbol{x}$, we can restrict the search for correspondences of that projection to the so-called epipolar line in the other image (red) rather than having to search the entire image.

between its epipolar plane and the image plane of the other camera. The correspondence search for $\tilde{\boldsymbol{u}}_R$ is now restricted to a line rather than the full image, with the position of $\tilde{\boldsymbol{u}}_R$ on the epipolar line influenced by the depth of $\boldsymbol{x}$ along the line from C_L to $\tilde{\boldsymbol{u}}_L$.

5.2.1.2 Rectifying the input images

Now we know that the corresponding point in the other image lies on the known epipolar line. For ease of notation and efficiency of the computations we now transform the input images so that these lines are the same and are horizontal, i.e. so that the correspondence for $\tilde{\boldsymbol{u}}_L$ lies on the same scanline in $\tilde{\boldsymbol{u}}_R$. This is done by projecting each of the input images onto a common image plane in such a way that the epipolar lines are horizontal, i.e. the viewing directions of the new virtual cameras should be orthogonal to the baseline. The result is illustrated in figure 5.8. We will not discuss the practicalities of this problem; it is well solved in existing literature, and the technical is of no immediate interest in the context of this chapter. We refer interested readers to the work by Loop and Zhang [Loop 1999].

5.2.1.3 Disparity and depth

When the cameras are calibrated and the images have been rectified, there is a straightforward relationship between the disparity shift of correspondences $\boldsymbol{u}_L$ and $\boldsymbol{u}_R$, and the depth value of the corresponding physical point $\boldsymbol{x}$. Figure 5.9 shows the projection of the epipolar plane on the horizontal plane. Through simi-

Figure 5.8: Example of the stereo rectification process. In the top row, the raw capture is shown – the images are slightly rotated and corresponding points are not on the same scanlines. This means that stereo matching algorithms need to perform a 2D search instead of a line search when looking for correspondences. In the bottom row, the images have been rectified based on a set of sparse correspondences (from the SIFT matching). It is now possible to view this image pair using the cross-eyed technique introduced earlier, and stereo matching algorithms now need only search on the same scanline for correspondences.

lar triangles, it becomes clear that the z-coordinate of the point, which is the same in either camera's reference system as they are rectified, is inversely proportionate to the disparity $u_L - u_R$: $z = fL/(u_L - u_R)$. So in order to estimate the depth of the input pixels, it suffices to estimate the disparities with their corresponding pixels in the other view.

5.2.1.4 Foundations of stereo matching

As discussed in the previous section, the problem of estimating the depth maps is equivalent to estimate the disparity fields: for each pixel in either image, the disparity shift to its correspondence in the other image is needed. Unfortunately, we often face issues: homogeneous areas cannot be matched unambiguously, and some points are only ever visible in a single image, either due to occlusions or because their correspondence falls outside of the image boundaries. These problems are illustrated in figure 5.10. The way the various stereo matching techniques handle these issues is one of their most discriminating traits.

Stereo matching is one of the most fundamental problems in computer vision: although the estimation of an accurate depth map poses several challenges, such as occlusions and textureless regions, it is a prerequisite for many applications such as, e.g., view synthesis and scene interpretation. Consequently, stereo matching has received much attention in literature. Therefore, we now give a comprehensive

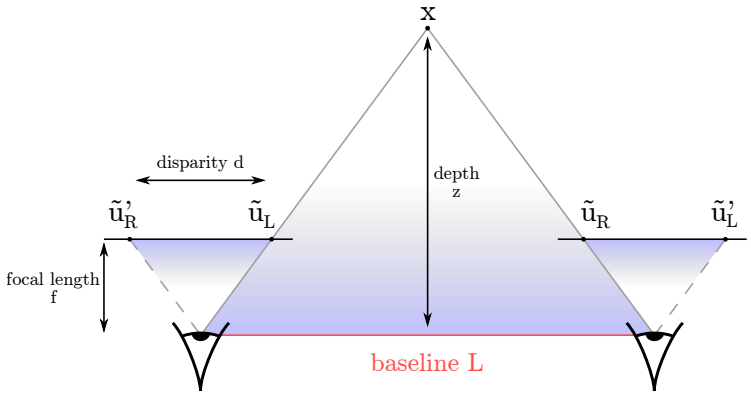


Figure 5.9: The depth in the cameras' coordinate system and the disparity ($u_L - u_R$) between both views are inversely related. This becomes clear when we consider the similar triangles highlighted in this figure.

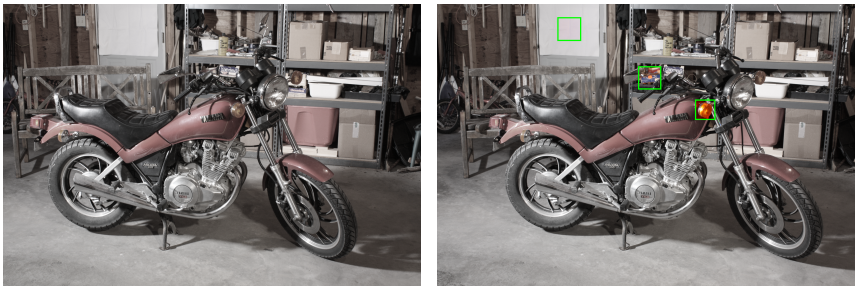


Figure 5.10: Some common issues with correspondence estimation highlighted in a sample from the Middlebury 2014 dataset [Scharstein 2014]. For the indicator light of the motorbike, the correspondence in the left image is rather obvious. However, for the white curtain on the back door, the correspondence is ambiguous and it is not clear exactly which patch corresponds to the same physical 3D point. Part of the rack behind the motorbike is hidden by the motorbike in the left image, and the correspondence is ambiguous.

overview of the existing work in this field.

We first introduce the four-step stereo matching pipeline as it was introduced by [Scharstein 2002]: matching cost calculation, cost aggregation, disparity calculation and finally disparity refinement. We list the most commonly used matching costs in literature together with their advantages and drawbacks. As far as the cost aggregation is concerned, we split up the literature in local and global methods, with a special mention for Semi-Global Matching [Hirschmüller 2008].

The traditional sum-of-squared-differences (SSD) method serves as a perfect example for this pipeline. First, the matching cost is computed as the SSD of the pixel correspondence hypotheses. Say that we have the set of possible depth hypotheses $\{d_i\}$. Then the cost $\phi(u, v, i)$ linked to the i'th depth hypothesis for pixel $[u, v]^{\mathrm{T}}$ in the left view is given by the difference between that pixel in the left view I_L and its hypothesized correspondence in the right view I_R, $[u + d_i, v]^{\mathrm{T}}$:

$$\phi(u, v, i) = \|I_L(u, v) - I_R(u + d_i, v)\|_2. \tag{5.2}$$

Assuming the dimension of the input image is $H \times W$, the result is a $H \times W \times D$ tensor containing the matching cost for each pixel for each of the D disparity hypotheses. In an aggregation step, these differences are averaged spatially to yield an aggregated cost volume $\Phi(u, v, i)$. To select the best disparity hypothesis $i^\star(u, v)$ for each pixel, we simply select the hypothesis with the lowest aggregated error: the Winner-Takes-All (WTA) solution:

$$i^\star(u, v) = \underset{i}{\operatorname{argmin}}\, \Phi(u, v, i). \tag{5.3}$$

In the following sections we give an overview of the various matching costs available to replace the basic formulation of equation (5.2), and the way the raw errors are aggregated before the best hypothesis is chosen. For the latter, we split the discussion in two: first, we discuss methods that work with a discrete set of disparity hypotheses (possibly refining it after selecting the optimal one), after which we discuss the variational methods which perform disparity optimization in the continuous domain. Lastly, we give an overview of recent work that replaces part of this pipeline, or even all of it, with machine learning approaches.

Matching cost choices

Stereo matching is in essence a correspondence problem; even though the search is strongly restricted by the epipolar lines, we nonetheless experience difficulties due to occlusions and homogeneous areas. The matching cost evaluates a given disparity hypothesis for pixel $\boldsymbol{u}$ - which ideally takes its lowest value at the true disparity.

However, the design of such a matching cost is hard. The most basic assumption made by stereo matching techniques is the so-called brightness constancy: if we assume that all objects in the scene are perfectly Lambertian, then a scene point

should have exactly the same appearance in all views [Oren 1995]. This leads to the absolute or squared differences as matching cost, as in equation (5.2). There are more robust cost functions that use truncated or mixed versions of these or other norms, such as the Huber loss which behaves quadratically at low errors and is linear at higher values, to be more robust against outliers [Huber 2011]. The use of a support window $\mathcal{N}(u, v)$ to compute the cost over, rather than a single pixel, often also helps:

$$\phi(u, v, i) = \sum_{(u_n, v_n) \in \mathcal{N}(u,v)} \|I_L(u_n, v_n) - I_R(u_n + d_i, v_n)\|. \tag{5.4}$$

This makes the implicit assumption that all pixels in the support window have the same disparity (i.e., that all surfaces are frontoparallel), although some methods have been proposed to use adaptive support regions [Zhang 2009].

More complicated cost functions have been proposed in literature, that use mutual information, the census transform, or entropy measures. However, real-world situations often violate the original brightness constancy assumption. For one, cameras with automatic or differing lighting settings cause exposure changes. More fundamentally, real cameras are also subject to vignetting, and the non-Lambertian behavior of real scenes also throws a spanner in the works. Hirschmüller and Scharstein call these issues radiometric changes, and have evaluated their impact on commonly used matching costs [Hirschmüller 2007]. Although the performance of a given matching cost obviously depends on the rest of the pipeline it is used in, these radiometric changes consistently degrade performance, and do so more for some matching costs than for others. For local, correlation-based methods, the rank filter [Zabih 1994] performed the best. For global methods, hierarchical mutual information [Hirschmüller 2005] won out for global illumination changes, while the rank filter outperformed it for local radiometric variations. However, none of the evaluated matching costs were able to handle strong lighting changes.

Local, global and semi-global methods

After defining a matching cost, we now wish to aggregate these costs before computing the disparity estimate, to apply some constraints or prior knowledge about the relationship between the disparities of nearby pixels. This aggregation and subsequent optimization is either global or local. Because local methods perform aggregation over a local support region, they cannot handle long-range interactions and often fail in cases that require complex reasoning, such as occlusions.

Global methods, on the other hand, formulate disparity computation as an energy minimization problem. The energy function is then split up into a data term, essentially the same as in local methods, and a regularizer term; often smoothness or piece-wise smoothness. For global models, the optimization graph is loopy in nature, and finding the global optimum is an NP-hard problem. A variety of gen-

eral optimization techniques provide means to compute approximate solutions: graph-cut, loopy belief propagation, or mean field approximations.

Alternatively, dynamic programming techniques can be used to find the global minimum of independent scanlines in polynomial time. However, handling each scanline completely independently from the others introduces noticeable streaking artifacts and loses vertical consistency. It is better to evaluate the cumulative cost function along several directions. Semi-Global Matching [Hirschmüller 2008] (SGM) is a global optimization algorithm based on this dynamic programming concept. The energy is calculated by summing costs along multiple lines through each pixel, oriented to the cardinal directions, after which the disparity estimate is selected using WTA. This approach has been very influential in literature, due to its combination of accuracy and speed.

Several follow-up works have further investigated the practical and theoretical aspects of SGM. Gehrig et al. [Gehrig 2009] proposed a real-time implementation of an extended version of SGM aimed at automotive applications. On the more theoretical side, Drory et al. [Drory 2014] discuss the success of SGM by clarifying its relation to message-passing with uncertainty measures as a belief-propagation optimization technique to the loopy graph problem. It has also recently been used to aggregate matching costs from CNN-learned features [Luo 2016, Žbontar 2016]. The performance can be improved even further by introducing confidence measures to the method [Seki 2016] or by also including a CNN for calculating the penalties at the global level [Seki 2017].

Variational methods for stereo matching

The aggregation methods from the previous section all work with a set of discrete disparity hypotheses. A distinct body of work within the stereo matching literature is given by the variational approaches. In these, the disparity field is optimized in the continuous domain using numerical methods. They work on top of the same matching cost data term $E_d(d(u,v))$, that can now be evaluated for arbitrary disparity hypotheses using linear interpolation, by imposing an additional regularization term E_s:

$$\Phi(d(u,v)) = \lambda E_d(d(u,v)) + E_s(d(u,v)). \tag{5.5}$$

We refer to section 8.3 for an in-depth discussion of the optimization approach.

The simple Total Variation (TV) smoothness term, which favors piece-wise constant regions, was used to good effect in problems such as noise suppression or pixel classification. However, it yields unconvincing results for stereo matching as it leads to stair-casing effects in areas with weak or ambiguous correspondences. Häne et al. introduce patch-based priors into a TV framework by using small, piecewise planar, dictionaries [Häne 2012]. Total Generalized Variation [Bredies 2010] (TGV) encourages piece-wise affine solutions rather than

only piece-wise constant ones. However, due to its complexity it is only applicable to convex data terms; for the original TV, in contrast, global solutions can be computed even in the case of non-convex data terms due to the strong regularizing effect. To preserve fine details, which tend to get lost in the hierarchical approaches required by variational approaches, Kuschk and Cremers integrate an adaptive regularization weight into the TGV framework based on edge detection [Kuschk 2013]. Ranftl et al. show how the issues with non-convex dataterms in combination with TGV can be alleviated by splitting the energy term [Ranftl 2013].

Deep learning for stereo matching

In recent years, deep learning approaches have gained much popularity in stereo matching. As mentioned before, some of these focus on learning point descriptors for use in the matching cost, or directly learn the matching cost between patches [Chen 2015, Luo 2016, Žbontar 2016]. While some methods have performed depth estimation from single images, we will focus this discussion on those that follow the stereo matching set-up: using two images as input to estimate disparity fields that imply the depth of the scene.

Mayer et al. propose Dispnet [Mayer 2016], by adapting the FlowNet encoder-decoder architecture from Dosovitsky et al. [Dosovitskiy 2015] that was so successful for optical flow estimation. The encoder computes abstract features while the decoder reestablishes the original resolution, all the while exploiting skip-connections within the network for fine-grained inference. However, such an approach does not expressly exploit the well-known geometric restrictions in stereo matching, and performs worse on the benchmarks than matching based approaches.

Seki and Pollefeys [Seki 2016] leverage a CNN to provide confidence predictions. By weighting each pixel according to its estimated confidence value, their proposed approach performs better in nearly-ambiguous regions. Seki and Pollefeys went on to directly learn the penalties for SGM [Seki 2017] with loss functions resembling the SGM energy function. While these two methods still start from the disparity hypothesis cost volume, derived from trained features in the case of [Žbontar 2016], Kendall et al. [Kendall 2017] form the cost volume using deep feature representations, which is subsequently filtered using a trained 3D network. Their model is trained end-to-end to regress disparities directly from the cost volume, providing an implicitly learned cost aggregation and disparity estimation procedure that fits well in pipeline model introduced earlier.

5.2.2 Light-field methods

In a more generalized version of binocular stereo, we take a look at the use of multiple cameras on the same baseline, rather than only two. This problem is situated between binocular stereo matching and general multi-view stereopsis (discussed

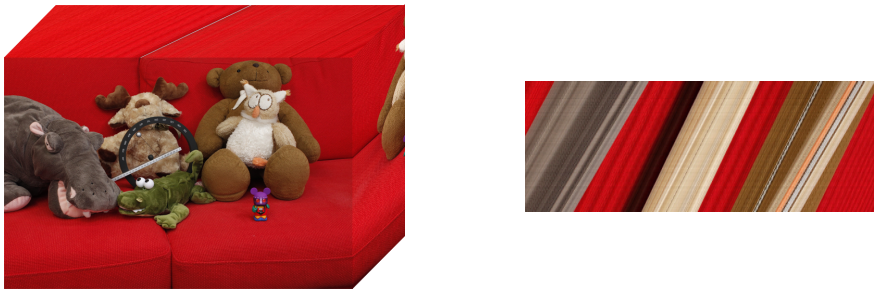


Figure 5.11: A visualization of the possible ways to slice a light-field to extract images. The lightfield is constructed by concatenating all the captured 2D images along a third axis, the camera axis (left). Slicing along this new axis, of course, yields us the original images. We can also slice the light-field along one of the image plane axes (right). In the corresponding light-field slices, also called epipolar slices, the angles of the lines indicate the depth of the corresponding pixels.

below), in which there is no constraint on the position of the cameras. Yet in this case, we can still leverage the existing work on binocular stereo matching. At the same time, the increased amount of information helps mitigate a typical issue binocular stereo suffers from: occlusions. In this case, we obviously have more information available: views from one side may have pixels occluded in them, but those pixels are unlikely to also be occluded in views on the other side.

Light-field cameras, as discussed in section 2.1.6, capture the scene from many cameras situated close to each other, on a baseline or on a rectangular grid. This essentially results in a 3D grid of intensity measurements [Levoy 1996], as illustrated in figure 5.11. Traditional viewing has us slice this grid along the camera axis, which results in extracting the various images as taken by the cameras. However, we can also slice this grid along on of the image axes: in that case, the so-called light-field slices or epipolar slices consist of a set of lines whose angles indicate the inverse depth of the corresponding pixels. The steeper the line is, the less that pixel's projected coordinates change with respect to a moving camera, and the further the corresponding 3D point is from the cameras.

As a result, depth can be inferred from such captures by estimating the angles of the lines [Kim 2013]. We argue in chapter 8 that the knowledge of binocular stereo matching algorithms can be directly mapped on this relatively new field [Donné 2015b], by explicitly taking occlusions into account. This has been concurrently exploited by other researchers [Tosic 2014, Wang 2015, Jeon 2015]. The drawback of these methods is the relatively large cost of the corresponding plenoptic cameras. As illustrated by [Kim 2013] and ourselves in the next chapter, a more budget approach is that of a camera dolly. Such a set-up is only useful for static scenes, as the cameras has to move over time to capture the various viewpoints, but it does allow for accurate depth estimates for such scenes.

The drawback for the camera dolly is that we do not always know the camera locations accurately, as we are often not able to control the dolly accurately. The camera locations can be estimated using a camera tracking solution based on SLAM or SFM techniques, such as the VOODOO method discussed before, but we illustrate in chapter 8 that leveraging the same stereo matching framework yields better results.

5.2.3 General Multi-view Stereopsis

General multi-view stereopsis (MVS) techniques make no more strong assumptions about the camera positions or their calibration. We typically assume that the cameras were intrinsically calibrated prior to the capture – a valid assumption, as those calibration parameters remain constant and as this can be done in a controlled setting.

As mentioned earlier, at this point we only consider methods that estimate a set of depth maps for the various views - other representations will be discussed in later sections. The depth map representation is typically preferred in scene analysis due to its flexibility and scalability to large scenes. One strategy that has proven to be particularly effective is the plane sweeping stereo algorithm [Collins 1996]. It creates a collection of parallel planes in a scene, projects all input images onto these planes using planar homographies, and then evaluates photo-consistency values on all planes to find the most likely occupied locations. For large scenes, one of the challenges is to handle the massive amount of data in real-time [Pollefeys 2008].

The current state-of-the-art is the Colmap method [Schönberger 2016b]. It combines previous improvements such as normal estimation [Galliani 2015], smart view selection [Bailer 2012], and smart occlusion handling [Shan 2014]. Additionally, a variety of photometric and geometric priors improve the robustness and accuracy of the reconstruction [Zheng 2014]. As a result, the method is applicable to extremely large reconstruction problems [Furukawa 2010b, Furukawa 2010a].

5.3 Volumetric methods

Volumetric approaches estimate the scene geometry on a regular 3D grid, in a variety of possible representations. The most basic representation is the discrete occupancy, which indicates for each volume element, or voxel, whether or not it is occupied [Kutulakos 2000]. For some applications, robotics in particular, it is more useful to encode a distance function on the grid, indicating for each voxel how far away it is from the surface [Faugeras 1998]. As a result, gradients can easily be calculated to move away from or towards the nearest object. More recent approaches use a probabilistic indicator of occupancy [Bhotika 2002].

The amount of memory required by the 3D grid, growing cubically, is the main limitation for volumetric approaches. Voxel hashing [Nießner 2013], or adaptive discretization in the form of delaunay triangulation [Labatut 2007] or octree representations [Steinbrücker 2014] can alleviate this issue.

A popular approach exploiting the same consistency assumptions made in stereo matching is the space carving method [Kutulakos 2000], in which voxels are considered occupied when their projection on the input images is photoconsistent. Its more simple version, shape-from-silhouettes [Baumgart 1974] or visual hull [Laurentini 1994], only takes a binary segmentation of the scene into account: assuming that the object of interest can be detected in the input images, we need only check for each voxel whether or not it is observed to be part of this foreground object. This binary segmentation is performed by so-called foreground-background segmentation algorithms. For more details, we refer the interested reader to the survey paper by Bouwmans [Bouwmans 2011].

As one can imagine, these volumetric techniques, apart from requiring much memory and storage, also require a large number of calculations. On the other hand, the voxels can be processed largely in parallel, making these techniques a prime candidate for implementation on a GPU. We discuss this further in chapter 9, when applying voxel carving to a practical problem.

In case depth maps are already available, we may want to fuse them into a single volumetric representation for easier reasoning. Curless and Levoy demonstrate that averaging truncated signed distance functions allows for a simple yet extremely effective approach to depth fusion [Curless 1996] that is still used by a large number of reconstruction pipelines today to combine their depth map estimates [Izadi 2011, Nießner 2013, Steinbrücker 2014].

5.4 Simultaneous Localization and Mapping

Simultaneous Localization and Mapping (SLAM) techniques address the problem of simultaneously reconstructing the scene structure and estimating the camera's position in that scene; the term is mostly used in the context of intelligent and/or autonomous vehicles. Many different SLAM techniques exist, many of which rely on similar techniques discussed earlier: we consider a two-dimensional axis for classifying the many techniques. The first discriminating trait is whether or not the variants have access to depth information; the second whether they match a sparse subset of the pixels or rather the entire input images.

SLAM methods that use depth information as input typically have the drawback that they are limited in operation range, due to the fact that their active measurement devices have a limited range. On the other hand, they are robust to the illumination changes and shadows that tend to confuse purely visual-based methods. Methods acting on only visual input often end up estimating depth maps

anyway as an intermediate step towards mapping [Engel 2014, Wang 2017]. Engel et al. estimate depth maps frame-to-frame, and use direct image alignment to estimate the frame-to-frame changes. They represent the 3D scene as a pose graph of keyframes with estimated depth maps [Engel 2014]. However, such frame-to-frame modeling is very susceptible to frame drift. As discussed later, this can be mitigated with loop closure approaches, but Wang et al. mitigate the issue by using a stereo camera set-up to avoid the drift problem [Wang 2017], as they can now estimate the depth maps for each frame independently (see section 5.2.1).

On the other axis, feature-based methods select a small subset of points, based on their distinctiveness and ease of tracking, to match and track between the images, and as a result they are typically faster. They use features like SIFT (Scale Invariant Feature Transform) [Lowe 1999], SURF (Speeded Up Robust Features) [Bay 2008], BRIEF (Binary Robust Independent Elementary Features) [Calonder 2010], FAST (Features from Accelerated Segment Test) [Rosten 2006], or ORB (Oriented FAST and Rotated BRIEF) [Rublee 2011] to perform point selection and matching. In this choice, there is an important trade-off: the features should be quick to compute but should also offer robust matching statistics. The current state-of-the-art for visual feature-based SLAM methods uses ORB features to do this, and with great success [Rublee 2011, Mur-Artal 2015].

Often, the localization and mapping are performed separately and in parallel [Klein 2007, Wang 2017]. The main idea here is that the mapping should reconstruct the scene as accurately as possible but does not necessarily need to use each input frame, nor does it need to restrict its calculation time to a single inter-frame interval. On the other hand, the localization approach should be fast enough to be done on each frame. Because of the previously mentioned drift problem in frame-to-frame matching (where small errors propagate and accumulate), the addition of a loop closure task is commonplace [Mur-Artal 2015], often in the shape of a global pose graph optimization [Kummerle 2011, Endres 2012, Kerl 2013, Engel 2014].

The algorithms using depth input typically have an easier time: they already start with a 3D reconstruction for each frame, albeit not aligned to the current world model, and are less susceptible to drift caused by frame-to-frame estimation of depth. Some techniques use semantic features like planes to speed up matching of large areas as well as to add semantic meaning to the reconstruction [Salas-Moreno 2014]. Other methods simply perform an alignment of this input cloud with the current scene model as the localization step, and expand the model with this point cloud [Endres 2012, Kerl 2013]. It is also possible to use both a frame-to-frame alignment and a frame-to-model alignment when the rough scene model is known in advance [Vlaminck 2017].

While all of these methods still create point cloud models of the scene, which grow in complexity through time, it is also possible to represent the scene as a

dense implicit volume in the form of a signed distance function. By doing this, Izadi et al. have a scene model whose complexity does not increase over time, and which can still be continously refined [Izadi 2011]. The largest drawback of their KinectFusion is that the scene size needs to be known in advance and is restricted due to memory constraints.

Sometimes, the main goal for applying SLAM techniques is actually only the localization part, for example for further use in the other 2D-to-3D reconstruction techniques discussed here. Several good commercial and non-commercial packages are available for this, such as the VOODOO camera tracking suite we compare with in chapter 8 [Thormählen 2006, Thormählen 2012] or the SFM part of the Colmap package [Schönberger 2016b].

5.5 Conclusions

In this chapter, we have given an extensive overview of the various techniques for performing 2D-to-3D reconstruction. By classifying the various techniques with respect to the sparseness and representation of their reconstruction, we have sketched an image going from coarse and sparse reconstructions to the dense representations by a set of depth maps and finally volumetric grids.

We have given relatively more attention to the problems of point triangulation, non-rigid structure-from-motion and stereo matching, as these will be discussed in more detail in the rest of this dissertation.

Having identified the main problem with point triangulation, which is the detection of outliers for correspondence searches in combination with the propensity for local minima of the ℓ_2 norm, we set out to speed up the point triangulation under the ℓ_∞-norm in chapter 6.

For non-rigid structure-from-motion techniques, which we wished to apply to on-the-fly reconstruction of faces in a teleconferencing setting, we have argued that existing fare badly in this setting. A shape model needs to be extracted from a constantly increasing set of past frames. Initial work has focused on iteratively expanding the shape basis as required, but this is not tenable for very long sequences. Instead, we explain in chapter 7 how to carefully curate a set of keyframes as history, which enables us to accurately and quickly extract a shape model from those.

Finally, we identify the two main issues with stereo matching techniques: high-resolution inputs and occlusions. First of all, and most simply, these methods are relatively costly and take a long time to estimate disparity for high-resolution images. By a carefully chosen low-resolution representation of the input images, it is possible to drastically speed up the execution times without a large loss of performance. On the contrary: in chapter 8, we show that selecting the right representation actually improved the noise robustness of the methods. Secondly, occlusions form a big issue for binocular stereo matching techniques.

Such occlusions have much less impact in light-fields, as we have more information available. However, the relatively new field of light-field imaging was not yet leveraging the extensive knowledge from stereo matching research. In the second part of chapter 8, we show how we have extended the binocular stereo matching formulation to multiple images on the same baseline, and how to estimate the position of the cameras along this baseline in an effective way, all the way carefully taking occlusions into account.

6

Efficient ℓ_∞ Point Triangulation

" But to have dreamed the dream is to have flown above the mountains so high in all but deed. "

— Peter F. Hamilton, *Judas Unchained*

As discussed in chapter 5, efficient point triangulation is an important aspect of outlier detection in point cloud reconstructions. Counter-intuitively, by quantifying the reconstruction error using the ℓ_∞-norm (and hence assuming that outliers do not exist), outliers will be apparent and easily detected after triangulation [Olsson 2010]. As an additional benefit, ℓ_∞ methods are not plagued by the local minima that can be present in traditional ℓ_2 methods [Hartley 1997, Byröd 2007, Hartley 2013] because of the pseudo-convex nature of the resulting ℓ_∞ cost function [Olsson 2007]. However, at the same time, preexisting ℓ_∞-norm based methods rely heavily on existing solver packages such as Secondary-Order Cone Problem solvers [Agarwal 2008, Kahl 2008b, Dai 2012], or the Sparse Bundle Adjustment package [Lourakis 2009, Eriksson 2014].

In this chapter, we present a rigorous geometric method that efficiently computes the optimum for the ℓ_∞-norm triangulation problem, without relying on complex solvers. While the CPU implementation is already noticeably faster than existing methods, the resulting algorithm is straightforward to implement on a GPU, which yields us an additional speedup. We start this chapter with a mathematical discussion of the triangulation problem and the various noise models that can be assumed. Afterwards we outline our contribution, the polyhedron collapse method, and compare it with the existing triangulation methods.

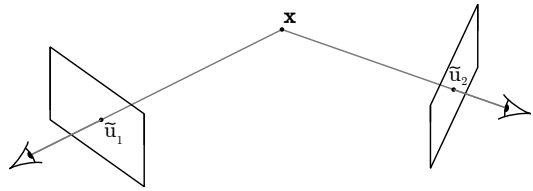


Figure 6.1: In the ideal case, we can estimate the 3D location as the intersection of all observation rays. However, in practical applications they never intersect at all, let alone in a single point, and we need to estimate the most likely point to have caused the observations in some other way.

6.1 The triangulation problem

Each camera $c \in \{1, \ldots, C\}$ is defined by an extrinsic rotation matrix R_c and a translation $\boldsymbol{t}_c$. We assume that the cameras were already calibrated intrinsically, so that the influences of the intrinsic matrix and possible lens distortions were already corrected for. As discussed in chapter 2, the relationship between a point's world coordinates $\boldsymbol{x}$ and its camera-specific coordinates $\boldsymbol{x}_c(\boldsymbol{x})$ is then given by the linear relationship

$$\boldsymbol{x}_c(\boldsymbol{x}) = \begin{bmatrix} x_c(\boldsymbol{x}) & y_c(\boldsymbol{x}) & z_c(\boldsymbol{x}) \end{bmatrix}^{\mathrm{T}} = R_c\boldsymbol{x} + \boldsymbol{t}_c. \tag{6.1}$$

Assuming a pinhole camera model [Hartley 2004], the point $\boldsymbol{x}$ is observed by camera c in the normalized pixel location $\boldsymbol{u}_c$:

$$\boldsymbol{u}_c(\boldsymbol{x}) = \begin{bmatrix} u_c & v_c & 1 \end{bmatrix}^{\mathrm{T}} = \begin{bmatrix} \frac{x_c}{z_c} & \frac{y_c}{z_c} & 1 \end{bmatrix}^{\mathrm{T}}. \tag{6.2}$$

In reality, due to noise in the observations and an imperfect camera calibration, camera c will rather observe the point at location

$$\tilde{\boldsymbol{u}}_c = \begin{bmatrix} \tilde{u}_c & \tilde{v}_c & 1 \end{bmatrix}^{\mathrm{T}} = \boldsymbol{u}_c(\boldsymbol{x}) + \boldsymbol{n}_c, \tag{6.3}$$

where each $\boldsymbol{n}_c$ is a noise vector with an unspecified distribution $p_n(\boldsymbol{n})$ (see section 6.2). As shown in figure 6.1, we can now try to invert the rays for these observations, and look for a point that is likely to have caused them. In the ideal case, the rays from all cameras intersect in a single point and our choice is obvious; more likely, due to the noise, they do not and it is not.

Then, given a hypothesis $\boldsymbol{x}$ for the point location, the reprojection error $\gamma_c(\boldsymbol{x})$ for camera c is given by

$$\gamma_c(\boldsymbol{x}) = \left\| \tilde{\boldsymbol{u}}_c - \boldsymbol{u}_c(\boldsymbol{x}) \right\|_q, \tag{6.4}$$

where q is a suitably-chosen power for the norm. Many methods in literature quantify the single view reprojection error in terms of the ℓ_2 norm, i.e. the Euclidean

distance, but as discussed earlier we will instead adopt the ℓ_∞-norm, also known as the max-norm.

In any case, triangulation methods minimize the aggregated reprojection error

$$\gamma(\boldsymbol{x}) = \|(\gamma_1(\boldsymbol{x}), \ldots, \gamma_C(\boldsymbol{x}))\|_p \,. \tag{6.5}$$

Here, too, the ℓ_2-norm is traditionally used to aggregate the error components. Instead, we have studied the use of the ℓ_∞-norm here, too, to aid the detection of outliers. In the next section we discuss the choice of the power-norms in equation (6.4) and equation (6.5) based on the assumed noise model.

In our proposed polyhedron collapse technique, we do not require a bisection algorithm, nor do we solve time-consuming auxiliary SOCP or BA problems, as do preexisting methods. Instead, the proposed algorithm directly minimizes $\gamma(\boldsymbol{r})$ through a sequence of line searches. The direction of the line search is computed in a straight forward manner conform the Karush-Kuhn-Tucker conditions, and the line searches only require us to solve some quadratic equations.

6.2 Noise models for point triangulation

Given equation (6.3), and assuming all noise instances are identically and independently distributed (i.i.d.), we can express the probability $p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right)$ of making a set of observations $\{\tilde{\boldsymbol{u}}_c\}$ given the point location $\boldsymbol{x}$ as

$$p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right) = \prod_{c=1}^{C} p_n(\tilde{\boldsymbol{u}}_c | \boldsymbol{x}_c(\boldsymbol{x})). \tag{6.6}$$

We now wish to estimate the most likely position $\boldsymbol{x}$, i.e. we wish to find the value of $\boldsymbol{x}$ that best explains our observations $\tilde{\boldsymbol{u}}_c$. This will depend on the noise distribution $p_n(\cdot)$, several possibilities for which are discussed below. Which of these is the best choice is application-specific: typically, Gaussian noise is a good model, but when the noise impact is generally low and outliers are rare, uniform noise is often a valid approximation too. As mentioned previously, assuming uniform noise helps in detecting outliers because they are not well explained by the uniform noise model.

6.2.1 Additive White Gaussian Noise - the ℓ_2 norm

In the case of AWGN, with zero mean and variance σ^2, we have

$$p_n(\tilde{\boldsymbol{u}}_c | \boldsymbol{x}_c(\boldsymbol{x})) = \frac{1}{2\pi\sigma^2} e^{-\dfrac{\|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_2^2}{2\sigma^2}}, \tag{6.7}$$

so that maximization of the whole likelihood from equation (6.6) yields

$$\begin{aligned}\underset{\boldsymbol{x}}{\operatorname{argmax}}\, p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right) &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \prod_{c=1}^{C} p_n(\tilde{\boldsymbol{u}}_c | \boldsymbol{x}_c(\boldsymbol{x})) \\ &= \underset{\boldsymbol{x}}{\operatorname{argmax}} \prod_{c=1}^{C} \frac{1}{2\pi\sigma^2} e^{-\frac{\|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_2^2}{2\sigma^2}} \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sum_{c=1}^{C} \|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_2^2.\end{aligned} \tag{6.8}$$

Rewriting this expression shows us that this is equivalent to the $p = q = 2$ triangulation problem:

$$\begin{aligned}\underset{\boldsymbol{x}}{\operatorname{argmax}}\, p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right) &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \sqrt{\sum_{c=1}^{C} \|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_2^2} \\ &= \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\|\left(\ldots, \|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_2, \ldots\right)\right\|_2 .\end{aligned} \tag{6.9}$$

6.2.2 Laplacian noise - the ℓ_1 norm

In the case of zero-mean Laplacian noise with scale b, we have

$$p_n(\tilde{\boldsymbol{u}}_c | \boldsymbol{x}_c(\boldsymbol{x})) = \frac{1}{4b^2} e^{-\frac{\|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_1}{2b}}, \tag{6.10}$$

which mutatis mutandis leads to the $p = q = 1$ triangulation problem:

$$\underset{\boldsymbol{x}}{\operatorname{argmax}}\, p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right) = \underset{\boldsymbol{x}}{\operatorname{argmin}} \left\|\left(\ldots, \|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_1, \ldots\right)\right\|_1 . \tag{6.11}$$

6.2.3 Uniform noise - the ℓ_∞ norm

Let us now assume that the noise is uniformly distributed, with zero mean but unknown (symmetrical) range γ. This means that

$$p_n(\tilde{\boldsymbol{u}}_c | \boldsymbol{x}_c(\boldsymbol{x})) = n(\tilde{u}_c - u_c(\boldsymbol{x}))\, n(\tilde{v}_c - v_c(\boldsymbol{x})), \tag{6.12}$$

with

$$n(x) = \begin{cases} \dfrac{1}{2\gamma}, & \text{if } |x| \leq \gamma \\ 0, & \text{elsewhere} \end{cases} \tag{6.13}$$

being the density function for the uniform distribution, parametrized with γ.

Once again, we reformulate the combined likelihood from equation (6.6):

$$\begin{aligned}
&p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right) \\
&= \prod_{c=1}^{C} p_n(\tilde{\boldsymbol{u}}_c | \boldsymbol{x}_c(\boldsymbol{x})) \\
&= \prod_{c=1}^{C} n\left(\tilde{u}_c - u_c(\boldsymbol{x})\right) n\left(\tilde{v}_c - v_c(\boldsymbol{x})\right) \\
&= \begin{cases} \left(\dfrac{1}{2\gamma}\right)^{2C}, & \text{if } \max\left(|\tilde{u}_c - u_c(\boldsymbol{x})|, |\tilde{v}_c - v_c(\boldsymbol{x})|\right) \leq \gamma, \ \forall c \\ 0, & \text{elsewhere.} \end{cases}
\end{aligned} \tag{6.14}$$

The measurements are assumed to have zero likelihood whenever any of the errors is larger than the range γ, but all other measurements are equally likely.

Now, we optimize over both $\boldsymbol{x}$ and the unknown γ to yield the $p = q = \infty$ triangulation problem:

$$\begin{aligned}
&\underset{\boldsymbol{x},\gamma}{\operatorname{argmax}}\, p\left(\tilde{\boldsymbol{u}}_1, \ldots, \tilde{\boldsymbol{u}}_C | \boldsymbol{x}\right) \\
&= \underset{\boldsymbol{x},\gamma}{\operatorname{argmax}} \begin{cases} \left(\dfrac{1}{2\gamma}\right)^{2C}, & \text{if } \|\tilde{\boldsymbol{u}}_c - \boldsymbol{u}_c(\boldsymbol{x})\|_\infty \leq \gamma, \ \forall c \\ 0, & \text{elsewhere.} \end{cases} \\
&= \underset{\boldsymbol{x}}{\operatorname{argmax}} \left(\frac{1}{2\| \left(\ldots, \|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_\infty, \ldots\right) \|_\infty}\right)^{2C} \\
&= \underset{\boldsymbol{x}}{\operatorname{argmin}} \|(\ldots, \|\boldsymbol{u}_c(\boldsymbol{x}) - \tilde{\boldsymbol{u}}_c\|_\infty, \ldots)\|_\infty
\end{aligned} \tag{6.15}$$

6.2.4 Mixed-norm triangulation problems

It is worth discussing the noise models implied by the mixed-norm triangulation problem, specifically for the combination norm $p = \infty$, which is sometimes used to avoid the local minima issues with $p = 2$. Similar to the above approaches, one can show that $(p = \infty, q = 1)$ and $(p = \infty, q = 2)$ are also maximum-likelihood estimators for different noise models: respectively, for a uniform distribution over a diamond rather than the square from section 6.2.3, and for a uniform distribution over a circle. This is illustrated in figure 6.2. Note that it is intuitively obvious that in 2D $q = 1$ is equivalent to $q = \infty$, after a rotation of the image plane coordinate systems (as the scaling does not affect the location of the optimum).

The uniform distribution over a circle has the advantage of being isotropic, i.e. being independent of the rotation of the coordinate system. However, at the point that the ℓ_∞ aggregation norm becomes a valid choice, we are typically already in a regime with low measurement noise, and other aspects such as quantification noise

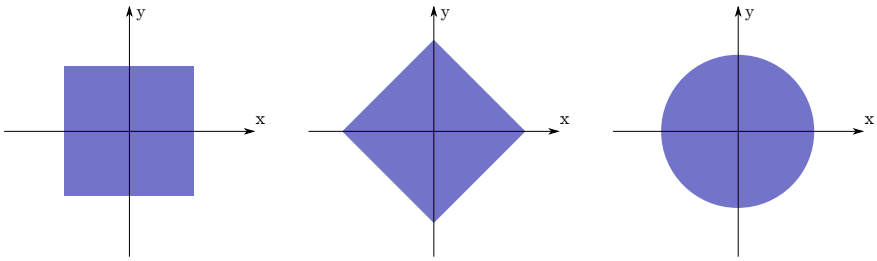


Figure 6.2: Isotope illustrations of various uniform noise distributions in the image plane. These lead to various ℓ_∞ aggregated cost functions, if we want to find the maximum-likelihood estimators of the underlying 3D point.

(due to the discretization of the pixel grid) become a bigger concern. Isotropy is then not necessarily a correct assumption anymore.

6.3 Point triangulation using polyhedron collapse

We now outline our proposed method for efficiently performing full-ℓ_∞-norm triangulation. By extension, as discussed in section 6.2.4, this is also applicable to $(p = \infty, q = 1)$-triangulation after the relevant transformations of the image plane coordinate systems.

We will adopt the common convention that only points in front of the cameras are of interest, i.e. that $z_c(\boldsymbol{x}) > 0, \forall c$: this is also called the cheirality constraint [Hartley 1993]. This is a purely physical restriction, after all: objects that are behind an image plane cannot possibly be imaged by that camera and so would never occur in the group of features defining the triangulation problem.

Recall that $\boldsymbol{x}_c(\boldsymbol{x}) = z_c(\boldsymbol{x})\boldsymbol{u}_c(\boldsymbol{x})$, so that we can rewrite the single-view reprojection error $\gamma_c(\boldsymbol{x})$ as

$$\gamma_c(\boldsymbol{x}) = \|z_c(\boldsymbol{x})\tilde{\boldsymbol{u}}_c - \boldsymbol{x}_c(\boldsymbol{x})\|_\infty / z_c(\boldsymbol{x}). \tag{6.16}$$

The objective function from equation (6.15) can be reformulated as

$$\min_{\boldsymbol{x}} \max_c \gamma_c(\boldsymbol{x}), \tag{6.17}$$

or equivalently as a constrained optimization problem with auxiliary variable γ:

$$\min_{\boldsymbol{x},\gamma} \gamma \text{ subject to } \gamma \geq \gamma_c(\boldsymbol{x}), \ \forall c. \tag{6.18}$$

In order to get rid of the non-linearities in the ℓ_∞-norm in equation (6.16), we introduce the constant vectors $\boldsymbol{i}_1 = (1, 0, 0)^{\mathrm{T}}$,$\boldsymbol{i}_2 = (-1, 0, 0)^{\mathrm{T}}$, $\boldsymbol{i}_3 = (0, 1, 0)^{\mathrm{T}}$,

and $\boldsymbol{i}_4 = (0, -1, 0)^{\mathrm{T}}$. Using these, we split up the set of C non-linear constraints for $\gamma_c(\boldsymbol{x})$ into a set of $4C$ linear constraints with auxiliary functions $\gamma_{c,l}(\boldsymbol{x})$:

$$\begin{aligned} & \min_{\boldsymbol{x},\gamma} \gamma \text{ subject to} \\ & \qquad \gamma \geq \gamma_{c,l} = \boldsymbol{i}_l \cdot (\boldsymbol{x}_c(\boldsymbol{x}) - z_c(\boldsymbol{x})\, \tilde{\boldsymbol{u}}_c) \,/ z_c(\boldsymbol{x}) \quad \forall c, l \\ \equiv \quad & (\gamma + \boldsymbol{i}_l \cdot \tilde{\boldsymbol{u}}_c) z_c(\boldsymbol{x}) \geq \boldsymbol{i}_l \cdot \boldsymbol{x}_c(\boldsymbol{x}) \quad \forall c, l. \end{aligned} \tag{6.19}$$

For a given value of γ, the four constraints of camera c define a half-open pyramid in space, topped at camera c's location. All points within this pyramid have a reprojection error on camera c that is smaller than γ, i.e. this pyramid is the feasible set for this camera's set of constraint. The feasible set for all of the constraints from equation (6.19) is given by the intersection of these pyramids for all cameras. As the intersection of a set of convex polyhedra, the result is also a convex polyhedron.

Our optimization algorithm iteratively reduces the value of γ until it approaches the optimal value $\gamma^\star$. Because of the pseudo-convexity of the ℓ_∞ problem shown by [Eriksson 2014], which implies that all sub-level sets are convex, we know that polyhedra for smaller values of γ are completely contained in the previous polyhedron. This also means that any point in the strict interior of the polyhedron is at least as good of a solution to the triangulation problem as its border points (which are all equivalent to the objective function). Therefore, the polyhedron corresponding to the optimal value $\gamma^\star$ has zero volume – typically it is a single point, but conceivably it can also be a line segment or a bounded part of a plane. If it had non-zero volume, then one of the interior points would be a better solution than the border points, and not all of the points in the set would be optimal.

This discussion also yields us the basic intuition behind our proposed approach: given an estimate $\boldsymbol{x}^{(t)}$, we compute an improving direction, based on local information, which points towards the interior of the polyhedron. We then perform a line search along this direction, which we show to boil down to solving a set of scalar quadratic equations. The only restriction for the initial estimate $\boldsymbol{x}^{(0)}$ is that it must fulfil the cheirality constraint previously mentioned: it must be visible to all cameras.

6.3.1 Choice of improving direction

Starting from an estimate $\boldsymbol{x}^{(t)}$, we implicitly construct the polyhedron for $\gamma(\boldsymbol{x}^{(t)})$. By necessity, $\boldsymbol{x}^{(t)}$ lies on the hull of this polyhedron so that at least one of the $4C$ constraints from equation (6.19) is active, i.e. fulfilled with equality. We now select an improving direction that steps away from all active inequalities for $\boldsymbol{x}^{(t)}$. As all of these constraints are planes, they are well represented by their gradients $\boldsymbol{g}_{c,l}(\gamma)$, the inward-pointing normals. These can be calculated from the linear

version of the $\gamma_{c,l}(\boldsymbol{x})$ constraint in equation (6.19):

$$\boldsymbol{g}_{c,l}(\gamma) = R_c^{\mathrm{T}} \left(-\boldsymbol{i}_l + (0, 0, \gamma + \boldsymbol{i}_l \cdot \tilde{\boldsymbol{u}}_c)^{\mathrm{T}}\right). \tag{6.20}$$

Assuming that there are J active constraints, we denote their inwards-pointing unit-norm normals by $\boldsymbol{n}_j$ in favor of brevity. We now wish to select an improving direction $\boldsymbol{d}^{(t)}$ that respects all of these gradients as much as possible, i.e. which maximizes the worst conformance as quantified by the inner product:

$$\boldsymbol{d}^{(t)} = \operatorname*{argmax}_{\boldsymbol{d}} \min_j \boldsymbol{n}_j \cdot \boldsymbol{d}^{(t)} \text{ subject to } \|\boldsymbol{d}^{(t)}\|_2 = 1. \tag{6.21}$$

It can be proven that the generalized solution (in more than three dimensions) is given by the unit vector pointing towards the center of the minimal enclosing sphere of the point set $\boldsymbol{n}_j$. For point triangulation, a three-dimensional problem, the optimal improving direction can be calculated more easily.

When $J = 1$, we simply have $\boldsymbol{d}^{(t)} = \boldsymbol{n}_1$. When the point lies on two faces of the polyhedron, and hence $J = 2$, we take the average direction:

$$\boldsymbol{d}^{(t)} \propto \boldsymbol{n}_1 + \boldsymbol{n}_2, \tag{6.22}$$

where we have omitted the normalization. This is simply the bisector of both planes in this point, see figure 6.3 for a 2D visualization. For three active constraints, things become a bit more complicated. Now, the minimal enclosing sphere could be defined by either two or three of the active constraints – typically three, but for (nearly) collinear triplets, there will only be two guiding points for the minimal enclosing sphere. In the case that all three active constraints span the minimum enclosing sphere, the optimal direction has the same inner product to each of the three normals, and can be constructed as:

$$\boldsymbol{d}^{(t)} \propto \boldsymbol{n}_1 \times \boldsymbol{n}_2 + \boldsymbol{n}_2 \times \boldsymbol{n}_3 + \boldsymbol{n}_3 \times \boldsymbol{n}_1, \tag{6.23}$$

where the sign is chosen to point towards the interior. This is the space bisector of the triangular pyramid with the tip at the origin and its edges given by the $\boldsymbol{n}_j$, and its inner products with each of the normals are all equal by virtue of the circular symmetry of the box product:

$$\boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 \times \boldsymbol{n}_3) = \boldsymbol{n}_3 \cdot (\boldsymbol{n}_1 \times \boldsymbol{n}_2) = \boldsymbol{n}_2 \cdot (\boldsymbol{n}_3 \times \boldsymbol{n}_1). \tag{6.24}$$

In order to avoid costly checks as to the coplanarity and to avoid having to construct the minimal enclosing sphere of the normal set, we simply calculate all three pair-wise directions as well and simply select the direction which maximizes the cost function in equation (6.21).

A similar approach can be used for $J > 3$. Because all our sample points lie on the unit sphere, the case of four active constraint dictating the unit sphere

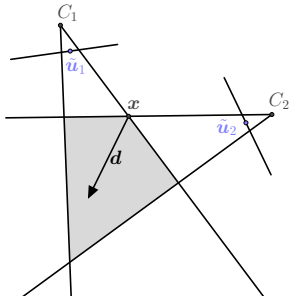


Figure 6.3: 2D visualization of the direction choice in the case of two active constraints. As there are two constraints active, the current estimate $\boldsymbol{x}$ lies on the intersection of two polyhedron faces, and we select the bisector as the locally optimal improving direction.

automatically means it coincides with the unit sphere. In that case, however, the optimal value of equation (6.21) is non-positive and no improving direction exists. Therefore, we need not evaluate that scenario. So any minimal enclosing sphere of interest is defined by either 2 or 3 points: we calculate all pair-wise and triplet-wise improving directions, and pick the one that yields the best value in equation (6.21). While the complexity of this step may appear to scale horribly, we show later in the results that the number of simultaneously active constraints is only very rarely higher than three or four, such that this scenario has minimal impact on the actual run time of our proposed algorithm.

We show in section 6.A that the algorithm fails to find an improving direction (i.e. the optimal value in equation (6.21) is non-positive) if and only if the Karush-Kuhn-Tucker conditions are fulfilled. This implies that the current estimate is a stationary point of the triangulation problem and, due to the pseudo-convex nature of this problem, is therefore a global optimum. In terms of the set of normals, this means that they are contradictory and that their minimally enclosing sphere coincides with the unit sphere centered at the origin.

6.3.2 Line search

We now follow the chosen direction $\boldsymbol{d}^{(t)}$ to a new estimate $\boldsymbol{x}^{(t)}$:

$$\boldsymbol{x}^{(t)} = \boldsymbol{x}^{(t-1)} + \alpha \boldsymbol{d}^{(t)}. \tag{6.25}$$

For notational brevity in the following discussion, let

$$f_{c,l}(\alpha) = \gamma_{c,l}\left(\boldsymbol{x}^{(t-1)} + \alpha \boldsymbol{d}^{(t)}\right) \tag{6.26}$$

be the (c,l)'th constraint value over the line. For all active constraints, $\frac{\partial}{\partial \alpha} f_{c,l}(0)$ is negative (as the inner product with the inwards-pointing normals is strictly pos-

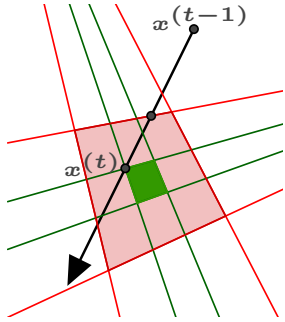


Figure 6.4: Example of the line search in two dimension. Starting in $\boldsymbol{x}^{(t-1)}$, we step towards $\boldsymbol{x}^{(t)}$ so that it again lies on an edge of the polyhedron (the polygon in this 2D example). The unlabeled intermediate point lies only on an edge of the polyhedron and is rejected: we can still take a step along the improving direction without an increase in the cost function. If we were to continue along the line, the polyhedron would expand again, the current master constraint would no longer dictate the value of γ and our local model is no longer meaningful.

itive, or the algorithm would have terminated already). We call the master constraint $(c^\star, l^\star)$ the one with the least negative derivative in 0: the constraint which changes the least and will hence remain active the longest for small steps along this line. Because this constraint is active, we know that $f_{c,l}(\epsilon) \geq f_{c^\star,l^\star}(\epsilon)$ for sufficiently small but positive ϵ. At some value $\alpha^{(t)}$, we will have $f_{c',l'}(\alpha^{(t)}) = f_{c^\star,l^\star}(\alpha^{(t)})$ for some (c', l'), beyond which point our current master constraint will no longer be dictating the value for $\gamma(\boldsymbol{x})$; we step exactly until this handover point. Such an intersection is sure to exist in the interior of the polyhedron, so that the new point $\boldsymbol{x}^{(t)}$ has a strictly lower reprojection error $\gamma(\boldsymbol{x}^{(t)})$ than our previous estimate. This is easily seen by evaluating the complementary constraint to the master constraint: the one with differing sign (i.e. $\boldsymbol{i}_1 \leftrightarrow \boldsymbol{i}_2$ and $\boldsymbol{i}_3 \leftrightarrow \boldsymbol{i}_4$) will always be reached as its value is equal when $f_{c^\star,l^\star}(\alpha) = 0$, such that there is a finite upper bound to α.

Geometrically, the selected value $\alpha^{(t)}$ is the lowest possible value for which $\boldsymbol{x}^{(t)}$ again lies on an corner of the polyhedron, as illustrated for two dimensions in figure 6.4. This implies that, except for the very first iteration, the improving direction will always be chosen based on at least two active constraints.

As $z_c(\boldsymbol{x})$ and $\boldsymbol{x}_c(\boldsymbol{x})$ are linear functions of $\boldsymbol{x}$ and $\tilde{\boldsymbol{u}}_c$ is a known constant vector, from (6.19) we can write that

$$\gamma_{c,l}(\boldsymbol{x}) = \frac{\boldsymbol{i}_l \cdot (\boldsymbol{x}_c(\boldsymbol{x}) - z_c(\boldsymbol{x})\, \tilde{\boldsymbol{u}}_c)}{z_c(\boldsymbol{x})} = \frac{\boldsymbol{a}_{c,l} \cdot \boldsymbol{x} + b_{c,l}}{\boldsymbol{c}_{c,l} \cdot \boldsymbol{x} + d_{c,l}}. \tag{6.27}$$

This in turn implies that solving $f_{c',l'}(\alpha^{(t)}) = f_{c^\star,l^\star}(\alpha^{(t)})$ for $\alpha^{(t)}$ boils down to solving a quadratic equation in $\alpha^{(t)}$.

6.3.3 Practical implementation

In practice, due to machine precision, some inequalities may only be roughly active:

$$\gamma(\boldsymbol{x}) \approx \gamma_{c,l}(\boldsymbol{x}). \tag{6.28}$$

We therefore evaluate whether constraints are active by using a threshold ϵ, i.e. by checking whether

$$(1-\epsilon)\gamma(\boldsymbol{x}) \leq \gamma_{c,l}(\boldsymbol{x}), \tag{6.29}$$

for which we have used $\epsilon = 10^{-5}$ in our implementation. Aside from that, we note here that the proposed approach, being so lightweight, is a prime candidate for implementation on the GPU. Each point in a large dataset can be reconstructed completely independently, and one could easily optimize all points in lock-step for a given number of iterations for a very efficient implementation at the cost of the per-point convergence criterion. We show the resulting timing comparison below.

6.4 Comparison to existing methods

As the proposed method finds the global optimum just as other existing methods in literature, the focus of the comparison goes to the speed with which the methods converge to this solution. A concern that has been raised earlier is that our approach needs to evaluate all possible pairs and triplets when the number of active constraints is larger than three. Therefore, we first show that this is very rare in practice; after that, we show that the proposed method indeed executes much faster than preexisting methods. We evaluate on several datasets: both the *dino* dataset [1] and the datasets of Alcatraz, Église du Dôme, Örebro Castle, Stockholm town hall and the Vercingetorix statue [Enqvist 2010, Olsson 2011].

6.4.1 The number of active inequalities

In the datasets by Olsson et al. [Enqvist 2010, Olsson 2011], a large number of cameras observe each of the points. As an example, the Örebro castle dataset is a collection of 761 views with a total of 58951 points, each visible in a subset of the views. Figure 6.5 visualizes how many cameras observe each point: this ranges from only a handful to over a hundred cameras. For this dataset, we have visualized the number of active inequalities through time in figure 6.6. The first iteration nearly always only has a single active inequality (the initial point is very likely to lie on a face of its corresponding polyhedron), but after that there are always at least two active inequalities. It is clear that four active inequalities is already pretty rare; in the entire Örebro dataset, there was only one single point

[1] Available on `www.robots.ox.ac.uk/~vgg/data/data-mview.html`, but not formally published.

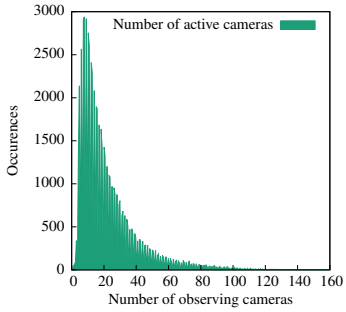


Figure 6.5: Histogram of the number of cameras observing each point in the Örebro dataset.

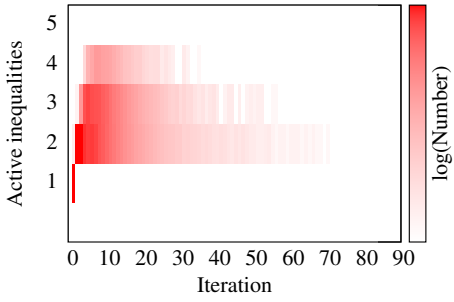


Figure 6.6: Heatmap (in log-scale) for the number of active constraints in a given iteration, normalized over the entire Örebro dataset.

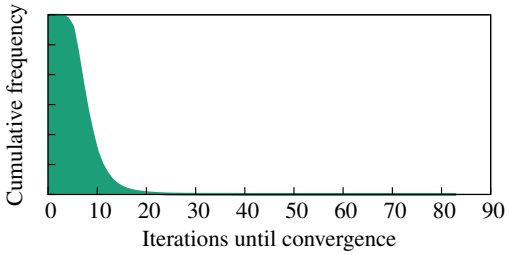


Figure 6.7: Distribution of the number of iterations required to reach convergence (over all points in the Örebro dataset).

that had five active inequalities at once. Finally, we also show the number of iterations required for convergence in figure 6.7. The proposed method converges in a relatively small number of iterations, each of which is extremely cheap, as it entails only the calculation of the improving direction and solving the set of separate quadratic equations for $f_{c',l'}(\alpha^{(t)}) = f_{c^\star,l^\star}(\alpha^{(t)})$.

6.4.2 Run time

We compare with the Gugat algorithm from Agarwal et al. [Agarwal 2008], both using the SOCP approach for the $(p = \infty, q = 2)$ problem and the LP approach for the $(p = \infty, q = 1)$ problem. The approach by Eriksson et al. [Eriksson 2014] serves as the state of the art for $(p = \infty, q = \infty)$. Figures 6.8 through 6.13 illustrate that our proposed method executes noticeably faster than the existing methods; the results are summarized in table 6.1. All of the techniques were evaluated in MATLAB using SeDuMi as the solver for the LP and SOCP problems [Sturm 1999], and SBA for bundle adjustment [Lourakis 2009].

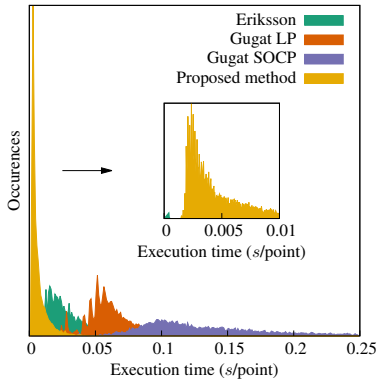


Figure 6.8: Run time for the discussed methods on the Alcatraz dataset with 65072 points over 419 views.

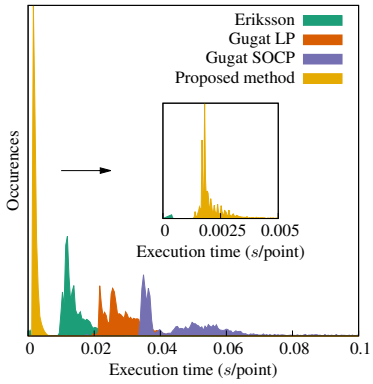


Figure 6.9: Run time for the discussed methods on the dino dataset with 4983 points over 36 views.

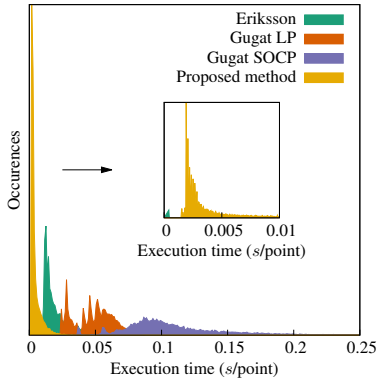


Figure 6.10: Run time for the discussed methods on the Église du Dôme dataset with 84792 points over 85 views.

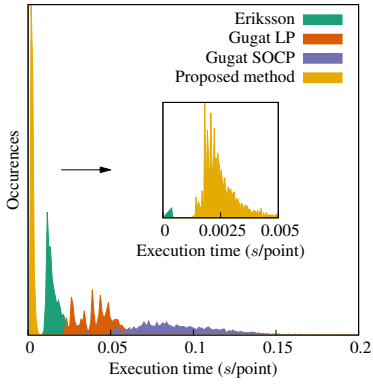


Figure 6.11: Run time for the discussed methods on the Örebro Castle dataset with 59856 points over 761 views.

	Synthetic	Alcatraz	Örebro	dino	Stockholm	Église	Vercingetorix
Gugat SOCP	109.0	254.0	133.8	47.5	156.1	161.5	113.1
Gugat LP	49.8	63.1	42.2	29.0	54.8	49.8	51.3
Eriksson	16.3	40.4	15.4	14.0	29.4	26.2	17.7
Proposed	4.5	6.6	2.5	2.2	5.2	4.4	2.6

Table 6.1: Average run time in milliseconds per point, over all evaluated datasets.

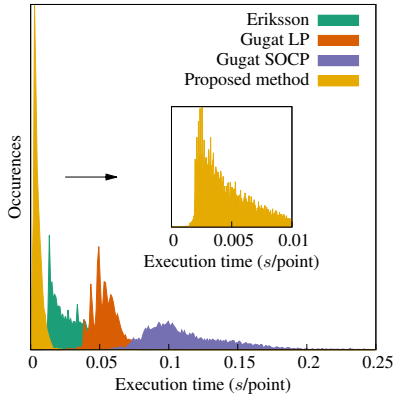


Figure 6.12: Run time for the discussed methods on the Stockholm Town Hall dataset with 28096 points over 43 views.

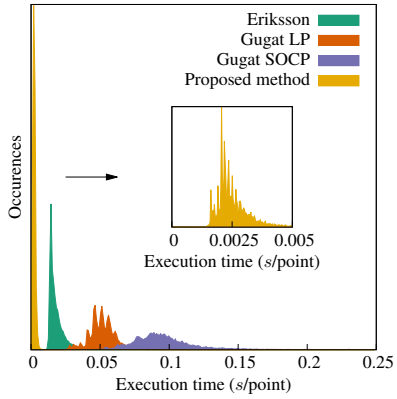


Figure 6.13: Run time for the discussed methods on the Vercingetorix dataset with 10789 points over 68 views.

6.4.3 Implementation on a GPU

The above timing comparison used a MATLAB implementation of the polyhedron collapse. Due to the straightforward nature of the proposed algorithm, we have no need to rely on complex solvers or preexisting software packages. To illustrate this, as well as illustrate once more the power of GPUs for parallel processing, we have implemented the same algorithm also within the Quasar framework [Goossens 2018]. For this experiment, one of the big advantages of Quasar is that it uses the same code-base for CPU and GPU execution: the Quasar code gets compiled into lower-level C++ or CUDA code, respectively. As a result, the comparison uses code that has had exactly the same amount of optimization for both cases. The results, shown in figure 6.14, illustrate that once again the GPU clearly outperforms the CPU on very large datasets.

Although this size of datasets (millions of points) is unlikely to occur often in practice, it is also useful to use this algorithm for outlier detection in which case many correspondence hypotheses need to be tested. The fact that this algorithm was implemented in a relatively short time (two working days by someone very familiar with both Quasar and the algorithm) highlights its potential for other systems, e.g., embedded devices, for which the complex optimization packages required by existing techniques do not exist.

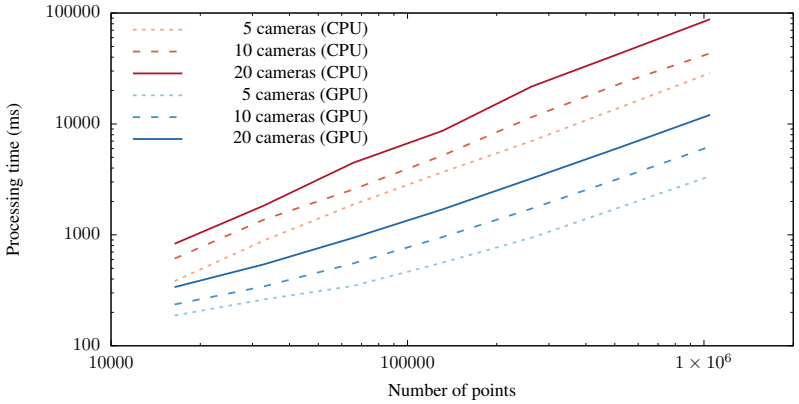


Figure 6.14: We have implemented our method on the GPU, in the Quasar framework, to process many points in parallel to exploit the GPU's parallel power. We ran 20 iterations of the polyhedron collapse algorithm on all points, regardless of convergence status or not, and report the results here. It is clear that the GPU boasts a big speed boost over the CPU implementation, up to a factor 10 on the larger tests, while there is a noticeable overhead for small datasets. For these experiments we have focused solely on processing speed: they were done on a a synthetic dataset with Gaussian noise and a fixed number of cameras per point.

6.5 Contributions

In this chapter, we have discussed the problem of efficient point triangulation under the ℓ_∞ norm. We have proposed an efficient geometrically-inspired optimization technique that is faster than existing techniques and simple to implement, even on GPUs.

A similar approach could be used to find the find the solution to the mixed-norm problems (with $p = \infty$). The workflow is identical, but the equations required to calculate the stepsize α are now no longer quadratic, but can be more involved depending on the choice of q.

The work contained in this chapter was published at a peer-reviewed international conference:

- *Point triangulation through polyhedron collapse using the ℓ_∞ norm*, Simon Donné, Bart Goossens, Wilfried Philips. International Conference on Computer Vision, ICCV 2015 [Donné 2015c].

Towards the future, we aim to apply this same optimization technique to other ℓ_∞ problems in multi-view geometry, such as camera resectioning or known-rotations reconstructions.

6.A The KKT criterion for ℓ_∞ point triangulation

We now prove that our proposed point triangulation algorithm under the ℓ_∞ norm terminates if and only if the KKT criterion is fulfilled. This implies that the current estimate is a stationary point, which – due to the pseudo-convexity – implies that it is a global optimum and that our algorithm was right to terminate.

We continue the notation introduced earlier in this chapter. The optimization problem we are addressing is

$$\begin{aligned} &\min_{\boldsymbol{x},\gamma} \gamma \text{ subject to} \\ &\qquad (\gamma + \boldsymbol{i}_l \cdot \tilde{\boldsymbol{u}}_c) z_c(\boldsymbol{x}) \geq \boldsymbol{i}_l \cdot \boldsymbol{x}_c(\boldsymbol{x}) \; \forall c, l, \end{aligned} \tag{6.30}$$

where $\tilde{\boldsymbol{u}}_c$ is the measurement in image c, $\boldsymbol{x}$ is the 3D location we are estimating and $x_c\boldsymbol{x}$ is the transformation of $\boldsymbol{x}$ into the coordinate system of camera c, dictated by its rotation matrix R_c and translation vector $\boldsymbol{t}_c$. The $4C$ linear constraints, i.e. planes, are well-represented by their unit-norm normals $\boldsymbol{n}_{c,l}$. At any time, J constraints are active, represented with normals $\boldsymbol{n}_j$. The Karush-Kuhn-Tucker criterion now states that

$$\begin{aligned} &\nexists \text{ improving direction} \\ &\quad \iff \exists \boldsymbol{\lambda} \in \mathbb{R}^J : \\ &\qquad\qquad \mathbf{0} = \sum_j \lambda_j \boldsymbol{n}_j \\ &\qquad \text{and} \\ &\qquad\qquad 1 = \sum_j \lambda_j \end{aligned} \tag{6.31}$$

We split the proof into four cases, based on the number of active inequalities, and handle each case separately. For each case, we wish to prove both necessity (if the KKT criterion is fulfilled, our algorithm terminates) and sufficiency (if our algorithm terminates, the KKT criterion is fulfilled) of our lemma. We can safely skip the case of a single active equation: in that case the polyhedron has non-zero volume and there is always an improving direction, and our algorithm never terminates; the KKT criterion is not violated in that case.

6.A.1 Two active inequalities

In this case, the termination criterion for our algorithm states that $\boldsymbol{n}_1 = -\boldsymbol{n}_2$.

Necessity

If the KKT criterion is fulfilled, we have:

$$\begin{aligned} \exists \lambda_1 \geq 0, \lambda_2 \geq 0 : \\ \mathbf{0} &= \lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2 \\ 1 &= \lambda_1 + \lambda_2. \end{aligned} \tag{6.32}$$

Without loss of generality, we assume $\lambda_1 > 0$. Then we can write the relationship between $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ as

$$\boldsymbol{n}_1 = \left(1 - \frac{1}{\lambda_1}\right) \boldsymbol{n}_2. \tag{6.33}$$

Recall now that $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$ are unit-norm vectors, so that the equality from equation (6.33) can only be fulfilled for $\lambda_1 = 1/2$ and $\boldsymbol{n}_1 = -\boldsymbol{n}_2$. Hence, our algorithm terminates.

Sufficiency

If $\boldsymbol{n}_1 = -\boldsymbol{n}_2$, then the choice $\lambda_1 = \lambda_2 = 1/2$ fulfills the KKT criterion.

6.A.2 Three active inequalities

In this case, the termination condition in the proposed algorithm is that the three gradients are coplanar and that none of the pairwise combinations are a valid option:

$$\begin{aligned} \boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 + \boldsymbol{n}_3) \leq 0 \\ \boldsymbol{n}_2 \cdot (\boldsymbol{n}_1 + \boldsymbol{n}_3) \leq 0 \\ \boldsymbol{n}_3 \cdot (\boldsymbol{n}_1 + \boldsymbol{n}_2) \leq 0 \end{aligned} \tag{6.34}$$

Necessity

From the KKT constraints we get

$$\begin{aligned} \mathbf{0} &= \lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2 + \lambda_3 \boldsymbol{n}_3 \\ &\implies \begin{cases} \lambda_1 \boldsymbol{n}_1 \cdot \boldsymbol{n}_2 = -\lambda_2 - \lambda_3 \boldsymbol{n}_2 \cdot \boldsymbol{n}_3 \\ \lambda_1 \boldsymbol{n}_1 \cdot \boldsymbol{n}_3 = -\lambda_3 - \lambda_2 \boldsymbol{n}_2 \cdot \boldsymbol{n}_3 \end{cases} \\ &\implies \lambda_1 \boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 + \boldsymbol{n}_3) = -(\lambda_2 + \lambda_3)(1 + \boldsymbol{n}_2 \cdot \boldsymbol{n}_3) \end{aligned}$$

In case $\lambda_1 = 0$, it follows that $\boldsymbol{n}_2 = -\boldsymbol{n}_3$ and $\boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 + \boldsymbol{n}_3) = 0$. If this is not the case, we see that $\boldsymbol{n}_1 \cdot (\boldsymbol{n}_2 + \boldsymbol{n}_3) \leq 0$. Mutatis mutandis for the two other inequalities from (6.34).

Sufficiency

For brevity, we will use the notation $\mu_{i,j} = \boldsymbol{n}_i \cdot \boldsymbol{n}_j$. Starting from the system

$$\begin{cases} \mu_{1,2} + \mu_{1,3} \leq 0 \\ \mu_{1,2} + \mu_{2,3} \leq 0 \\ \mu_{1,3} + \mu_{2,3} \leq 0 \end{cases} \tag{6.35}$$

We see that at most one of $\mu_{1,2}$, $\mu_{1,3}$ and $\mu_{2,3}$ can be positive. Without loss of generality, assume that $\mu_{2,3} \leq \mu_{1,3} \leq \mu_{1,2}$ and thus $\mu_{2,3} \leq 0$ and $\mu_{1,3} \leq 0$.

Case 1: $\mu_{2,3} = -1$
We can construct the required convex combination as $\frac{1}{2}\boldsymbol{n}_2 + \frac{1}{2}\boldsymbol{n}_3 = \boldsymbol{0}$.

Case 2: $\mu_{2,3} \in \ ]-1, 0]$
We will now construct a convex combination of the three gradients that equals the null vector.

$$\begin{aligned} &\lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2 + \boldsymbol{n}_3 = \boldsymbol{0} \\ \implies &\begin{cases} \lambda_1 + \lambda_2 \mu_{1,2} = -\mu_{1,3} \\ \lambda_1 \mu_{1,2} + \lambda_2 = -\mu_{2,3} \end{cases} \end{aligned} \tag{6.36}$$

The way in which this system was procured is valid if and only if the constructed convex combination of gradients is not orthogonal to both $\boldsymbol{n}_1$ and $\boldsymbol{n}_2$. However, for this to be the case (because the three gradients are coplanar) these two vectors have to be parallel. However, this is not possible as $\mu_{1,2} \in \ ]-1, 0]$.

Solving the system (6.36), we arrive at:

$$\begin{bmatrix} \lambda_1 \\ \lambda_2 \end{bmatrix} = \frac{1}{1 - \mu_{1,2}^2} \begin{bmatrix} 1 & -\mu_{1,2} \\ -\mu_{1,2} & 1 \end{bmatrix} \begin{bmatrix} -\mu_{1,3} \\ -\mu_{2,3} \end{bmatrix}$$

Both coefficients are positive. We prove this for the first coefficient:

$$\lambda_1 = \frac{1}{1 - \mu_{1,2}^2} \left(\mu_{1,2}\mu_{2,3} - \mu_{1,3}\right)$$

From $\mu_{1,3} + \mu_{1,2} \leq 0$ and $\mu_{2,3} \leq \mu_{1,3}$ we know that

$$\begin{aligned} &0 \leq (\mu_{2,3} - \mu_{1,3})(\mu_{1,3} + \mu_{1,2}) \\ \implies &\mu_{2,3}\mu_{1,2} \geq \mu_{1,3} + \mu_{1,3}(\mu_{1,3} - \mu_{2,3} + \mu_{1,2} - 1) \end{aligned}$$

Now, because $\mu_{1,3} \leq 0$, $\mu_{1,3} + \mu_{1,2} \leq 0$ and $\mu_{2,3} \geq -1$, we know that the second term on the right hand side is positive:

$$\mu_{2,3}\mu_{1,2} \geq \mu_{1,3} \implies \lambda_1 \geq 0$$

Mutatis mutandis we can prove the sign of λ_2.

It is now straightforward to construct the convex combination:

$$\frac{1}{1+\lambda_1+\lambda_2}\left(\lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2 + \boldsymbol{n}_3\right) = \mathbf{0}$$

6.A.3 Four active inequalities

In case four inequalities are active, the proposed algorithm terminates if there is a triplet that fulfills that causes termination (per the rules for three active inequalities), or if

$$\begin{cases} \boldsymbol{n}_1 \cdot \boldsymbol{d}_{234} \leq 0 \\ \boldsymbol{n}_2 \cdot \boldsymbol{d}_{134} \leq 0 \\ \boldsymbol{n}_3 \cdot \boldsymbol{d}_{124} \leq 0 \\ \boldsymbol{n}_4 \cdot \boldsymbol{d}_{123} \leq 0 \end{cases} \tag{6.37}$$

In which $\boldsymbol{d}_{ijk}$ equals the improving direction constructed from the triplet $\{i, j, k\}$ as

$$\begin{aligned} \boldsymbol{d}'_{ijk} &= \boldsymbol{n}_i \times \boldsymbol{n}_j + \boldsymbol{n}_j \times \boldsymbol{n}_k + \boldsymbol{n}_k \times \boldsymbol{n}_i \\ s_{ijk} &= (\boldsymbol{n}_1 \cdot \boldsymbol{d}'_{ijk}) / \left\| \boldsymbol{n}_1 \cdot \boldsymbol{d}'_{ijk} \right\| \\ \boldsymbol{d}_{ijk} &= s_{ijk} \boldsymbol{d}'_{ijk} \end{aligned}$$

in case the triplet is not coplanar, or as the average of two of the triplet's vectors if these vectors are coplanar such that the scalar product of any of the triplet's vectors with $\boldsymbol{d}_{ijk}$ is strictly positive.

Necessity

In case one of the $\lambda_i = 0$ (without loss of generality, assume λ_4) the proof reduces to the proof given in the case of three active constraints.

Therefore, we can assume for the rest of this section that $\lambda_i > 0,\ \forall i$. We rewrite one of the elements of the KKT criterion as follows:

$$\begin{aligned} &\mathbf{0} = \lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2 + \lambda_3 \boldsymbol{n}_3 + \lambda_4 \boldsymbol{n}_4 \\ &\quad \Longrightarrow \\ &0 = \lambda_1 (\boldsymbol{n}_1 \cdot \boldsymbol{d}_{123}) + \lambda_2 (\boldsymbol{n}_2 \cdot \boldsymbol{d}_{123}) + \lambda_3 (\boldsymbol{n}_3 \cdot \boldsymbol{d}_{123}) + \lambda_4 (\boldsymbol{n}_4 \cdot \boldsymbol{d}_{123}) \end{aligned} \tag{6.38}$$

As the construction of $\boldsymbol{d}_{123}$ differs depending on the coplanarity of the constituting vectors, we split up into the two possibilities.

The triplet $\{1, 2, 3\}$ is coplanar

In this case, $\boldsymbol{d}_{123}$ equals the average of two of the vectors in the triplet $\{1, 2, 3\}$.

Without loss of generality, assume that $\boldsymbol{d}_{123} = \frac{1}{2}\boldsymbol{n}_1 + \frac{1}{2}\boldsymbol{n}_2$. From (6.38) we arrive at:

$$\implies \lambda_4 \, \boldsymbol{n}_4 \cdot \boldsymbol{d}_{123} = -\frac{1}{2}(\lambda_1 + \lambda_2)(1 + \boldsymbol{n}_1 \cdot \boldsymbol{n}_2) - \frac{1}{2}\lambda_3(\boldsymbol{n}_1 \cdot \boldsymbol{n}_3 + \boldsymbol{n}_2 \cdot \boldsymbol{n}_3)$$

The two vectors from $\{1, 2, 3\}$ are chosen such that $\boldsymbol{n}_1 \cdot \boldsymbol{n}_3 + \boldsymbol{n}_2 \cdot \boldsymbol{n}_3 \geq 0$ and therefore the right hand side is non-positive. As $\lambda_4 > 0$ we conclude that $\boldsymbol{n}_4 \cdot \boldsymbol{d}_{123} \leq 0$.

The triplet $\{1, 2, 3\}$ is not coplanar
Continuing from (6.38) we arrive at:

$$\implies \boldsymbol{n}_4 \cdot \boldsymbol{d}_{123} = -(\lambda_1 + \lambda_2 + \lambda_3)\frac{\|\boldsymbol{n}_1 \cdot \boldsymbol{d}_{123}\|}{\lambda_4}$$

This step is possible because $\boldsymbol{n}_1 \cdot \boldsymbol{d}'_{123} = \boldsymbol{n}_2 \cdot \boldsymbol{d}'_{123} = \boldsymbol{n}_3 \cdot \boldsymbol{d}'_{123}$. This follows from the properties of the cross product and the triple product. As the right hand side is negative, so is $\boldsymbol{n}_4 \cdot \boldsymbol{d}_{123}$.

Mutatis mutandis we prove the other inequalities in (6.37). □

Sufficiency

If any of the triplets fulfill the termination requirements of a triplet, the algorithm terminates. Say, without loss of generality, that the triplet $\{1, 2, 3\}$ fulfills the requirements. In that case we can choose $\lambda_4 = 0$ and select λ_1 through λ_3 according to the procedure for three active constraints. We will therefore assume throughout the rest of this section that none of the triplets fulfill the termination requirements on their own.

At least one triplet is coplanar
This is not possible: in this case the quadruplet fulfills its constraints only if a coplanar triplet fulfills its constraints. If a triplet is coplanar and does not fulfill its constraints, all of its vectors lie in the same half-plane and hence we can express one of them as a positively weighted combination of the other two.

Without loss of generality, say the triplet $\{1, 2, 3\}$ is coplanar and that we can express $\boldsymbol{n}_3 = \kappa_1\boldsymbol{n}_1 + \kappa_2\boldsymbol{n}_2$, $\kappa_i \geq 0$. This results in a conflict:

$$\boldsymbol{n}_3 \cdot \boldsymbol{d}_{124} = \kappa_1\boldsymbol{n}_1 \cdot \boldsymbol{d}_{124} + \kappa_2\boldsymbol{n}_2 \cdot \boldsymbol{d}_{124}$$

This expression should be negative because of (6.37), yet the right hand side is a strictly positively weighed sum of strictly positive values. Hence, it is not possible for a triplet to be coplanar and for the quadruplet to still fulfill its constraints without that triplet fulfilling its constraints as well. This situation cannot occur.

None of the triplets is coplanar
We construct a convex combination as combination of the requirements

$$\mathbf{0} = \lambda_1 \boldsymbol{n}_1 + \lambda_2 \boldsymbol{n}_2 + \lambda_3 \boldsymbol{n}_3 + \lambda_4 \boldsymbol{n}_4$$

and

$$1 = \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 \tag{6.39}$$

From the first equality we derive four equalities by taking the scalar product of both sides with the improving directions of each of the possible triplets. This results in the following system:

$$\begin{bmatrix} \boldsymbol{n}_1 \cdot \boldsymbol{d}_{234} & \boldsymbol{n}_2 \cdot \boldsymbol{d}_{234} & \boldsymbol{n}_3 \cdot \boldsymbol{d}_{234} & \boldsymbol{n}_4 \cdot \boldsymbol{d}_{234} \\ \boldsymbol{n}_1 \cdot \boldsymbol{d}_{134} & \boldsymbol{n}_2 \cdot \boldsymbol{d}_{134} & \boldsymbol{n}_3 \cdot \boldsymbol{d}_{134} & \boldsymbol{n}_4 \cdot \boldsymbol{d}_{134} \\ \boldsymbol{n}_1 \cdot \boldsymbol{d}_{124} & \boldsymbol{n}_2 \cdot \boldsymbol{d}_{124} & \boldsymbol{n}_3 \cdot \boldsymbol{d}_{124} & \boldsymbol{n}_4 \cdot \boldsymbol{d}_{124} \\ \boldsymbol{n}_1 \cdot \boldsymbol{d}_{123} & \boldsymbol{n}_2 \cdot \boldsymbol{d}_{123} & \boldsymbol{n}_3 \cdot \boldsymbol{d}_{123} & \boldsymbol{n}_4 \cdot \boldsymbol{d}_{123} \end{bmatrix} \boldsymbol{\lambda} = \mathbf{0}$$

Per the construction of the improving direction, the scalar product of a gradient and the improving direction of any of the triplets is strictly positive. We divide each of the rows by the corresponding scalar product of the improving direction and one of the vectors in its triplet. The resulting system is:

$$\begin{bmatrix} \dfrac{\boldsymbol{n}_1 \cdot \boldsymbol{d}_{234}}{\boldsymbol{n}_2 \cdot \boldsymbol{d}_{234}} & 1 & 1 & 1 \\ 1 & \dfrac{\boldsymbol{n}_2 \cdot \boldsymbol{d}_{134}}{\boldsymbol{n}_1 \cdot \boldsymbol{d}_{134}} & 1 & 1 \\ 1 & 1 & \dfrac{\boldsymbol{n}_3 \cdot \boldsymbol{d}_{124}}{\boldsymbol{n}_1 \cdot \boldsymbol{d}_{124}} & 1 \\ 1 & 1 & 1 & \dfrac{\boldsymbol{n}_4 \cdot \boldsymbol{d}_{123}}{\boldsymbol{n}_1 \cdot \boldsymbol{d}_{123}} \end{bmatrix} . \boldsymbol{\lambda} = \mathbf{0}$$

Denote the entry on the diagonal for the i^{th} row as ϕ_i. We now subtract each of the rows from equation (6.39) in order to arrive at

$$\begin{bmatrix} 1-\phi_1 & 0 & 0 & 0 \\ 0 & 1-\phi_2 & 0 & 0 \\ 0 & 0 & 1-\phi_3 & 0 \\ 0 & 0 & 0 & 1-\phi_4 \end{bmatrix} . \boldsymbol{\lambda} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

From (6.37) it follows that $\phi_i \leq 0,\ \forall i$. The required convex combination is acquired by normalization of $\lambda_i = \frac{1}{1-\phi_i}$.

6.A.4 More than four active inequalities

In the case that more than four inequalities are active, the proposed algorithm terminates if there is a triplet that fulfills that causes termination (per the rules for three active inequalities) or if for each triplet the corresponding improving direction has a negative scalar product with at least one of the gradients.

Necessity

If exactly three λ_i or non-zero, the corresponding triplet fulfills the constraints as discussed in the case of three active constraints. We therefore only need to handle the case of more than three of them being non-zero.

Starting from the expression in the KKT criterion that

$$\sum_i \lambda_i \boldsymbol{n}_i = \mathbf{0}, \lambda_i \geq 0,$$

We split the summation for each triplet τ and take the inner product with $\boldsymbol{d}_\tau$:

$$\sum_{i \notin \tau} \lambda_i \boldsymbol{n}_i \cdot \boldsymbol{d}_\tau = -\sum_{i \in \tau} \lambda_i \boldsymbol{n}_i \cdot \boldsymbol{d}_\tau, \lambda_i \geq 0$$

The right hand side is non-positive: each gradient has a strictly positive scalar product with the improving direction of any triplet it belongs to and the weights are non-negative.

The left hand side, on the other hand, has at least one strictly positive weight (as more than three λ_i are non-zero). The left hand side must be non-positive, and all of its weights are non-negative (with at least one strictly positive weight). As a result, one of the terms being weighed must be non-positive.

Sufficiency

We show that it is possible to reduce this case to the case of four active inequalities: it is always possible to choose four inequalities such that those four also fulfill their constraints.

In order to show this, we iterate over all triplets τ. We then construct a matrix A as in the procedure for four active inequalities so that

$$(A_{\tau,i}) = \boldsymbol{n}_i \cdot \boldsymbol{d}_\tau$$

The matrix B is constructed by concatenating a row of 1s at the bottom:

$$B = \left[\frac{A}{\mathbf{1}}\right]$$

The addition of the last row has increased the rank at most by 1. The rank of B is therefore at most 4. Assume that the set $S_N = \{\boldsymbol{n}_1, \boldsymbol{n}_2, \boldsymbol{n}_3, \boldsymbol{n}_4\}$ spans the column space of B (as each column corresponds to one of the gradients, and each row corresponds to one of the triplets). Construct the matrix B_4 by selecting the four columns corresponding with S_N and the four rows corresponding with all possible triplets in this quadruplet in addition to the final row. Now we need only show that $\operatorname{rank}(B_4) = \operatorname{rank}(B)$.

We can do this by showing that the set S_T of improving directions for the triplets formed from the gradients in S_N spans the set Ω_T of all improving directions from the possible triplets of Ω_N, the set of all gradients for the active inequalities.

Geometrically, the improving directions are the central lines (bisectors) of the cones defined by triplets formed from S_N. If the vectors in S_N are not coplanar (and hence, span $\mathbb{R}^3$), then the central lines are not coplanar either and hence also span $\mathbb{R}^3$. For the planar option, if the vectors are not collinear, the central lines are not collinear and span the plane. If all vectors are collinear, then the improving directions all lie on the same line and hence span this line.

6.A.5 Conclusion

Combining all the above proofs for each case and sub-case, we have shown that our algorithm terminates if and only if the KKT criterion is fulfilled, and we have therefore arrived at a global optimum (due to the pseudo-convexity of the optimization problem).

7

On-line Non-Rigid Structure-from-Motion

" If you are not free to say no, your yes is meaningless. "

— Brent Weeks*, The Blinding Knife*

In chapter 5, we have discussed the various possibilities for sparse 3D reconstruction: the estimation of the 3D location of a relatively small set of points. Aside from single-point triangulation, which ignores temporal coherence of point clouds, there are two extensive fields of research for 2D-to-3D reconstruction of point clouds: Simultaneous Localization and Mapping (SLAM) techniques and Structure-from-Motion (SFM), both of which jointly estimate the 3D structure of the scene and the camera positions. SFM techniques assume that the point correspondences are known, i.e., that we know for each observation which physical 3D point it represents, while SLAM does not make this assumption and estimates these correspondences at reconstruction-time. On the other hand, SLAM techniques do not handle deforming scenes well. They are oriented towards the reconstruction of rigid scenes, and handling moving objects such as cars or people is typically done by filtering those objects. While traditional SFM techniques also cannot handle deforming scenes, Non-Rigid Structure-from-Motion (NRSFM) (an extension of SFM) is able to cope with a strongly deforming scene by modeling objects in a low-dimensional manifold and reconstructing their 3D structure through time.

In this chapter, we focus on the 3D reconstruction of faces in a teleconferencing scenario. Faces have a rich range of expressions, and hence it is important to

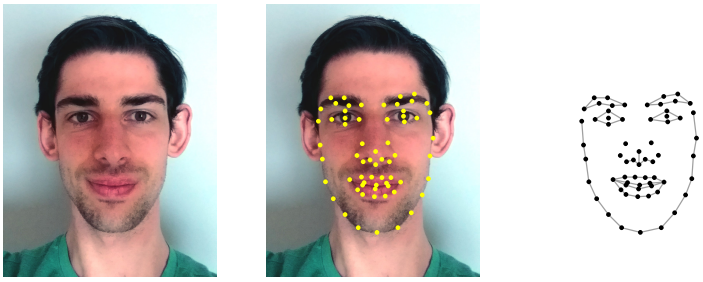


Figure 7.1: An example of the features extracted by a face detector and feature extractor. On the input image (left), various salient points are detected (middle). After the detection, we know which detection corresponds with which point in the face model (right), so that we have the correspondence info for the NRSFM methods.

be able to model strongly deforming objects. On the other, the assumption that point correspondences are available is more easily satisfied in this case: facial landmark detectors work well, especially in a front-view setting [Zhang 2010], and yield us the correspondences as a result of the detection procedure, see figure 7.1. Many NRSFM methods make the assumption that the camera is orthogonal; for the reconstruction of faces, this is a relatively good approximation, as they have a relatively small range of depth (less than 10 cm) compared to the distance to the camera (50 cm or more). And while NRSFM methods restrict the deformations of the objects to a low-dimensional manifold, this has been shown to be adequately expressive for representing the structure and deformation of faces [Blanz 1999, Wang 2008, Paysan 2009].

However, preexisting methods do not handle on-line scenarios well. NRSFM techniques extract a shape model before modeling each of the separate frames, and most of the methods extract it from the entire video sequence, after capture. Early research efforts into on-line approaches attempted to map existing off-line methods onto on-line methods, by iteratively expanding the shape basis [Paladini 2010, Tao 2013]. However, they have the propensity to gather noisy shape bases over time, causing them to over-fit to the input sequence. Instead, we have developed an approach that does not expand the shape basis iteratively, but rather stores a compact representation of the past frames and periodically extracts a new shape basis from those. Our proposed method is based on the Probalistic Principal Component Analysis (PPCA) [Torresani 2008], which was chosen for its good performance and high robustness to noise.

We first discuss the PPCA method in detail, and then outline our contribution: representing the history with a keyframes set from which to extract a shape basis. We show that the proposed approach does not lose much accuracy compared to a method that has access to all of the previously seen frames, while executing much faster and with a constant memory footprint.

7.1 NRSFM with Probabilistic PCA

Let $\boldsymbol{x}_{c,n}$ represent the 3D coordinates of point n in frame c, $\boldsymbol{t}_c$ that camera's translation and R_c its rotation matrix. We introduce the camera scaling factor s_c which is required for the approximation of a perspective camera by our orthographic model. As discussed earlier in chapter 2, s_c should be close to the inverse Z values of all points in the scene, in order for the approximation to be valid. The observation $\boldsymbol{u}_{c,n}$ of point n in frame c is then given by

$$\boldsymbol{u}_{c,n} = s_c R_c z \left(\boldsymbol{x}_{c,n} + \boldsymbol{t}_c\right) + \boldsymbol{n}_{n,c}. \tag{7.1}$$

We now concatenate these relationships for all points into a single matrix equation by stacking them vertically

$$\underbrace{\begin{bmatrix}\cdots & \boldsymbol{u}_{c,n}^{\mathrm{T}} & \cdots\end{bmatrix}^{\mathrm{T}}}_{U_C} = P_c \underbrace{\begin{bmatrix}\cdots & \begin{bmatrix}\boldsymbol{x}_{c,n}\\ 1\end{bmatrix}^{\mathrm{T}} & \cdots\end{bmatrix}^{\mathrm{T}}}_{X_c} + N_c, \tag{7.2}$$

where the entries of N_c are additive white Gaussian noise: $\boldsymbol{n}_{c,n} \sim \mathcal{N}(0; \sigma^2 \mathrm{I})$, and P_c is a $3N \times 4N$ block-diagonal matrix with copies of $s_c \left[R_c \mid \boldsymbol{t}_c\right]$ on its diagonal. We introduce the shape basis assumption by expressing X_c as an element of a K-dimensional manifold:

$$X_c = \bar{X} + V \boldsymbol{z}_c, \tag{7.3}$$

where $\bar{X}$ is the mean 3D shape, and V contains the shape basis vectors in its columns weighted by the per-frame deformation coefficients $\boldsymbol{z}_c$. At this point, Torresani et al. impose a Gaussian prior on the deformation weights, the principle behind PPCA: $\boldsymbol{z}_c \sim \mathcal{N}(0; \mathrm{I})$. Note that, when estimating the unknowns, the latent coordinates $\boldsymbol{z}_c$ are marginalized out: they are never explicitly solved for. As a linear transformation of a Gaussian variable is Gaussian, the distribution of the point cloud projection U_c is given by

$$U_c \sim \mathcal{N}\left(P_c \bar{X}; P_c V V^{\mathrm{T}} P_c^{\mathrm{T}} + \sigma^2 I\right). \tag{7.4}$$

Now, in order to estimate P_c, $\bar{X}$, V and $\boldsymbol{z}_c$, we optimize the joint likelihood of all measurements using the expectation-maximization algorithm [Dempster 1977]. The expectation-maximization (EM) algorithm alternates between two steps: in the E-step, a distribution over the latent coordinates $\boldsymbol{z}_c$ is computed; in the M-step the other variables are updated.

The E-step

Here, the posterior distribution over the latent coordinates given the current parameter estimates is computed, for each c. As mentioned earlier, this distribution

$q(\boldsymbol{z}_c)$ is Gaussian and can be expressed in closed form:

$$\begin{aligned} q(\boldsymbol{z}_c) &= p\left(\boldsymbol{z}_c | U_c, P_c, \bar{X}, V, \sigma^2\right) \\ &= \mathcal{N}\left(\boldsymbol{z}_c | \beta\left(U_c - P_c\bar{X}\right); I - \beta P_c V\right), \text{ where} \\ \beta &= V^{\mathrm{T}} P_c^{\mathrm{T}}\left(P_c V V^{\mathrm{T}} P_c^{\mathrm{T}} + \sigma^2 I\right)^{-1}. \end{aligned} \tag{7.5}$$

Given this distribution, we also define the following expectations:

$$\begin{aligned} \boldsymbol{\mu}_c &= E\left[\boldsymbol{z}_c\right] = \beta\left(U_c - P_c\bar{X}\right), \text{ and} \\ \phi_c &= E\left[\boldsymbol{z}_c \boldsymbol{z}_c^{\mathrm{T}}\right] = I - \beta P_c V + \boldsymbol{\mu}_c \boldsymbol{\mu}_c^{\mathrm{T}}. \end{aligned} \tag{7.6}$$

The M-step

In the M-step, the motion parameters are updated by minimizing the expected negative log-likelihood

$$E\left[\sum_{c=1}^{C} -\log p\left(\boldsymbol{u}_c | P_c, \bar{X}, V, \sigma^2\right)\right] \tag{7.7}$$

over P_c, $\bar{X}$, V and σ^2. This function, sadly, cannot be minimized in closed-form but is optimized by using coordinate descent. Closed-form updates can be computed for each of the individual parameters. For the explicit update rules, we refer to [Torresani 2008].

7.2 On-line NRSFM with a keyframe representation of history

In practice, the most difficult part of the non-rigid structure-from-motion techniques is the extraction of an accurate and complete shape basis. This is typically done based on all available frames; and in the case of an on-line setting this means all past frames. However, it is intractable to use all previous frames, as that number grows indefinitely throughout the reconstruction process. Therefore, we store a representative subset of past frames and discard the others. We periodically update the shape basis model based on this history representation, and then perform the reconstruction for the new frame based on the shape model. First, we give the overview of our complete workflow, and show how the previously discussed PPCA method fits in. Afterwards, we compare with the other on-line algorithms to show that our reconstruction results do not lose much accuracy, while requiring only a constant amount of memory storage.

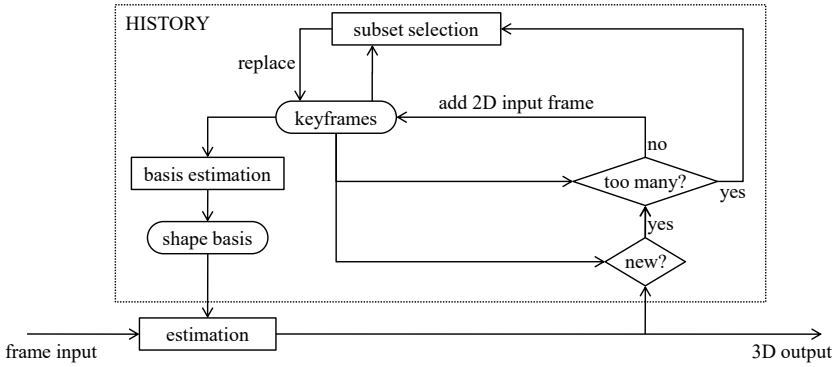


Figure 7.2: Flowchart of our proposed method for online non-rigid structure-from-motion. Using a keyframe representation of history, we estimate a shape basis model that helps us extract the camera locations and point cloud reconstructions. Whenever an unfamiliar point cloud pose is discovered, that image's input set of observations is added to the keyframe representation of history. Whenever the history grows too big, we cull the history to a smaller size that still covers all of the observed behaviors of the object.

7.2.1 Overview of our proposed method

We start by performing 3D reconstruction of a bootstrap sequence with an existing, off-line, reconstruction method. For this bootstrap process any of the previously managed techniques could be used, but we have again chosen the PPCA method for the same reasons as above: its good performance and high robustness to noise.

Then, we select a subset of frames from the bootstrap sequence which accurately represents the whole of the bootstrap as completely as possible, as outlined in the next section. As we process the rest of the video, we continuously model the input using this shape basis, extracting the camera location and deformation coefficients without the need for optimizing over the shape model itself. Whenever we encounter a frame which can not be adequately explained by the current shape model (the probability under the current shape model gets too small), it is added to the history. After this, a new shape model is extracted that serves as the new estimate for the shape basis. After a while, when the keyframe set has grown to a predefined threshold, we select a new subset of the currently-too-large keyframe set to serve as the new compact history representation. An overview of the method is given in the form of a flowchart in figure 7.2.

7.2.2 History representation with keyframes

The goal of the keyframe selection procedure is to select a set of frames which represent all possible deformations of the objects from as many vantage points as possible. Intuitively, assuming that we can represent a keyframe as a point in Eu-

clidean space, our goal is to select a subset whose elements are roughly uniformly sampled within the bounds of the full set. We investigate the use of both the deformation coefficients $\boldsymbol{z}_c$ and the camera parameters P_c to express the various frames in the Euclidean space. Note, however, that we placed a Gaussian prior on those coefficients, and are now aiming to cover the full set uniformly. This is intentional: while the 3D modeling of the face wants to penalize very extreme deformations and hence penalizes high deformation coefficients, we wish to explicitly include as much information about the object as possible into the history and extreme positions yield a lot of information that is not visible in other frames.

The number of ways to select a subset from the full set grows combinatorially with their sizes — an exhaustive comparison of all possible subsets is implausible. Instead, we resort to a heuristic subset selection. Starting from the full set, we remove one element at a time according to a local criterion. We evaluate the use of four different decision rules:

- maximizing the minimum distance between any two points in the subset,
- maximizing the mean distance between all elements of the subset,
- maximizing the mean distance from the elements in the subset to those of the full set, and
- maximizing the differential entropy of the subset (to estimate the differential entropy, we use a MATLAB wrapper for TIM, an open-source C++ library for efficient estimation of information-theoretic measures).

The result of each of these strategies on the selection of a representative subset for an artificial 2D point cloud is shown in figure 7.3. We can see that maximizing the minimum distance between any two points in the subset yields the most representative cloud, while maximizing the mean distances represents only the outer edges well. Maximizing the entropy metric yields disappointing results, which we assume to be caused by the difficulty of estimating the entropy of a continuous variable based on a relatively small number of samples.

Results shown in figure 7.4 demonstrate the impact of the subset selection criterion on the resulting reconstruction. The red baseline is given by the shape basis extraction from the full set of frames, a total of 316. We call the subset of keyframes representative of the full set if the reconstruction error is not significantly affected by restricting the estimation of the shape basis to the set of keyframes rather than allowing the use of the full set. The other curves show the relative reconstruction error to this baseline when using a subset of all frames as selected by three different metrics. We use the Euclidean distance between deformation coefficients as the distance metric in shape space, and we use the spatial angle between camera orientations to serve as the distance in the camera manifold.

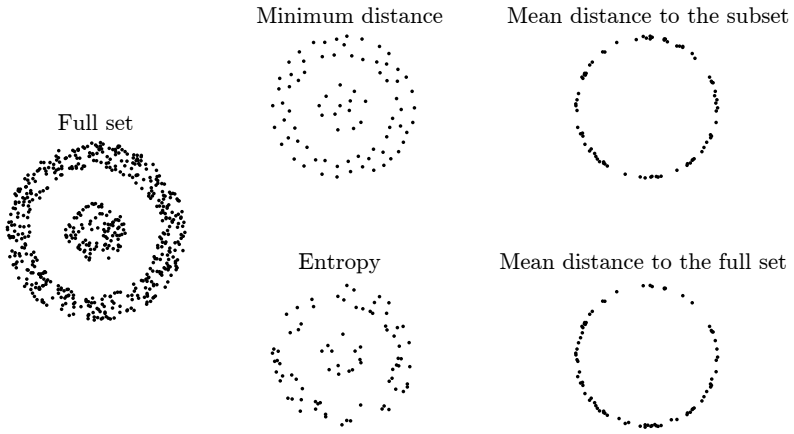


Figure 7.3: Demonstration of the subset selection metrics on an artificial 2D point cloud. Maximizing the minimum distance between any two points of the resulting point set gives the most visually pleasing results. Maximizing the mean distance, between either any two points of the resulting set or between the points of the resulting set and those of the original set, only represents the outer edge of the point cloud well, as expected. The results for the greedy maximization of the entropy of the resulting set are disappointing: we assume that the entropy measure is too unstable for such a small number of points.

However, in the combined shape-camera space, there is no straightforward approach for a unified Euclidean distance. Therefore we re-use the entropy selection criterion mentioned above. Figure 7.4 shows the comparison for three different metrics:

- maximizing the minimum distance in the shape space,
- maximizing the minimum distance in the camera manifold, and
- maximizing the entropy in the combined shape-camera space.

This figure shows that the minimum distance on the shape manifold results in a lower reconstruction error than the other methods, but most importantly that all of the metrics are able to reconstruct the scene well even when the shape basis is only estimated from a small set of frames – a strong indicator that our proposed approach is valuable and that a meaningful shape basis can be extracted from a small number of keyframes.

7.2.3 Comparison with other methods

We now present the result of the proposed approach and compare it to preexisting state-of-the-art methods. It is important to note that our method is restricted to causal processing of the input sequence: we reconstruct frame t using only the

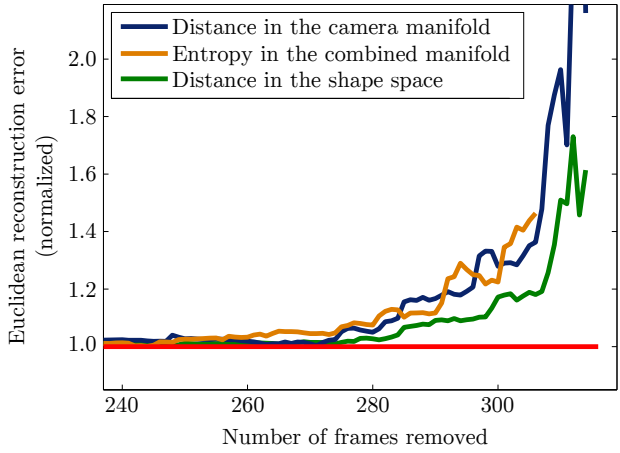


Figure 7.4: Keyframe selection using several metrics. The input sequence is a 316-frame sequence of a person talking [Torresani 2008]. To the right side of the graph, shape basis estimation is performed based on ever smaller subsets of the original frames. Up until a subset size of 35*, reconstruction is nearly as good as using the full video as input (between* 4% *and* 10% *worse). After that, performance does deteriorate noticeably.*

observations from frames 1 through t. The existing methods work off-line on the whole video sequence and are not restricted by causality: for a more fair comparison we restrict these to causal reconstruction as well, by extracting a new shape basis for each frame based on all previous frames. In this way, we focus our evaluation on the effectiveness of our history representation and mitigate the effect of the increased information, especially in the earlier frames. Clearly, this is not a practical way to perform reconstruction, but it gives an indication of the performance of other methods in a causal setting, and specifically the causal version of PPCA should yield an upper bound for the performance of our method. Figure 7.5 shows the comparison between the proposed method and two preexisting state-of-the-art methods [Gotardo 2011a, Torresani 2008]. While the proposed method can never perform a reconstruction of the same accuracy as its off-line variant, because it does not retain the same amount of information on previous frames, our proposed method still compares favorably to the off-line PPCA method and outperforms the Column Space Fitting method by a noticeable margin.

These graphs of course do not reflect the immense advantage of the on-line aspect of our method. There is no entry for the first online method by [Paladini 2010] because it uses a threshold for the error as an indicator of whether or not the model should be expanded. The resulting line in the graph is therefore constant and does not reflect the downsides of the method (extraordinary computational complexity with rising noise). The graph was produced by perturbing the 316-frame sequence from [Torresani 2008] with zero-mean Gaussian noise (AWGN), reconstructing

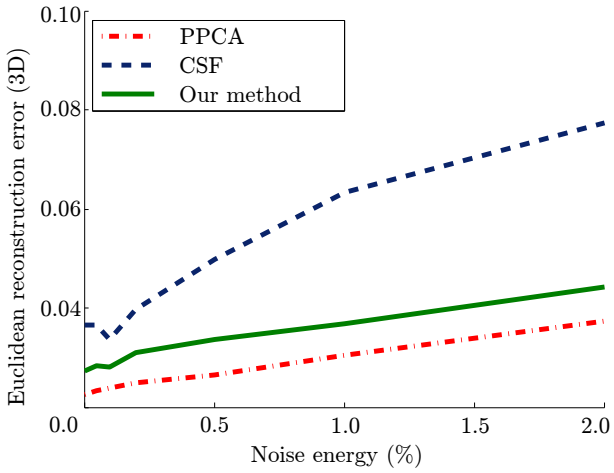


Figure 7.5: Causal comparison between two existing methods, PPCA [Torresani 2008] and CSF [Gotardo 2011a], and our proposed method. The input sequence consists of a person talking, courtesy of [Torresani 2008]. Noise energy is the fraction of noise standard deviation over the length of the diagonal of the ground-truth 3D bounding box.

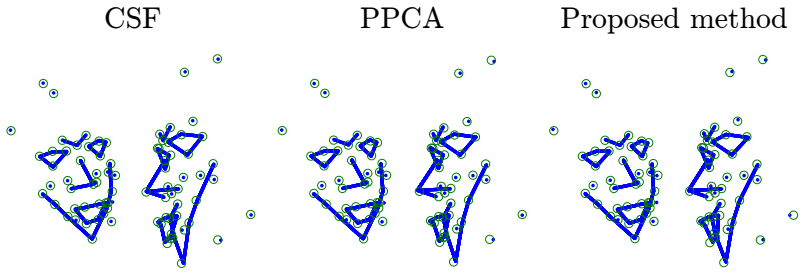


Figure 7.6: Visual comparison between three methods. Each time, the left face shows the original view point and the right row shows an alternate view point from the side. Lines and points show the projection of reconstructed 3D points and their connections, while small circles are centered around the ground truth locations.

the perturbed sequence with the various methods and afterwards computing the average 3D error between the reconstructed point cloud and the ground truth. The specific parameter values for our method were: a 60-frame bootstrap length, a 40-frame history representation size and 25 EM iterations per frame.

Figure 7.6 puts these reconstruction errors into perspective: even with the difference in reconstruction error, all three methods still manage an accurate reconstruction of the face from the sequence from [Torresani 2008]. We also present a reconstruction by the three methods of the sequence extracted from Elephant's Dream in Figure 7.7. For this comparison, we extract the projected points from the

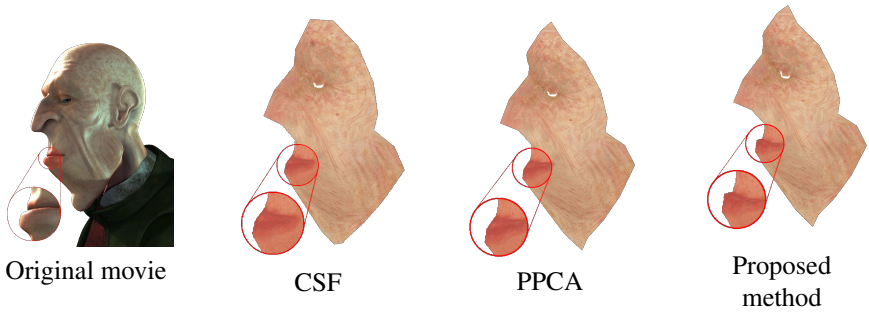


Figure 7.7: Visual comparison between two existing methods and our proposed method on a sequence extracted from Elephant's Dream.

video's source and pass them as input to the reconstruction methods. Because the full mesh of the Proog's face has over 4000 points, we manually selected a subset of 197 points to perform reconstruction on. After reconstruction, the resulting 3D mesh is visualized by using the original texture from the movie sources.

7.3 Contributions

In this chapter, we have proposed an on-line NRSFM approach, which efficiently represents the history of the previously seen frames in a small set of frames and discards all the others. From this history, we periodically extract a shape model that we use to reconstruct the arriving frames. In contrast to the existing methods, we are not constantly extending our shape basis (which would accumulate noise over time by adding noisy entries to the shape basis and never removing them), nor do we restrict our memory to just the shape basis, as would be the case with progressive PPCA (which may have issues forgetting rare base elements over time). Rather, we completely recalculate the shape basis from the measurements as required, whenever our history representation has changed: the size of the shape basis is fixed, and long-ago frames are as important as recent ones. As a result, we are able to process videos of unlimited length without the issues that other methods have: either having to rely on a constantly growing history or having to constantly expand the shape model and thus over time accumulate noise.

The work in this chapter was published at a peer-reviewed international conference:

- *Online non-rigid structure-from-motion based on a keyframe representation of history*, Simon Donné, Ljubomir Jovanov, Bart Goossens, Wilfried Philips, Aleksandra Pižurica. International Conference on Computer vision Theory and Applications, VISAPP 2014 [Donné 2014].

In the future, we envision two things. First of all, the combination of the low-resolution reconstructions we have created here with high-resolution face meshes, to animate them realistically in real-time. Secondly, and more fundamentally, we would like to study the possibility of updating the PPCA model in an on-line manner. For standard principal component analysis such techniques exist, and recently more robust approaches have been proposed that can better cope with input noise [Feng 2013]. However, for the on-line update of a PPCA model, more research is required.

8

Efficient and Multi-view Stereo matching

" The direct use of force is such a poor solution to any problem, it is generally employed only by small children and large nations. "

— David Friedman

Stereo matching techniques estimate dense depth fields for their input images. Assuming that the input images are rectified as discussed in section 5.2.1.2, this is done by finding the disparity for each pixel: the horizontal offset between itself and its correspondence in the other image (which is guaranteed to lie on the same row in each image). As a result of the rectified set-up, the depth of a pixel is inversely proportional to this disparity.

In chapter 5, we have identified several challenges for stereo matching techniques. First of all, there is simply the manner of input resolution: high-resolution inputs require more calculations, even though many of those might be redundant as neighboring pixels often have the same or very similar depth. Secondly, scene points are often occluded in either of the views, so that the depth of these pixels needs to be inferred in another way: through regularization. In this chapter, we show how to address both of these problems.

In order to do so, we extend an existing variational stereo matching technique. First, we discuss this technique's model, its assumptions and its execution in detail. Using this existing approach as a starting platform, we have made two contributions to the literature: robust and more efficient processing of high-resolution

images, and the extension of binocular stereo matching to light-field set-ups. By leveraging an over-segmentation of the input images, we manage to exploit both the efficiency of the low-resolution representation and the noise-robustness of the over-segmentation technique to estimate the disparity and depth on the higher resolution inputs. Secondly, by extending the binocular stereo matching approach to a light-field set-up, we can leverage the extensive literature from stereo matching, which among others has had extensive studies on occlusion handling, to improve performance of depth estimation in such scenarios.

8.1 Variational stereo matching

We first discuss in detail the variational stereo matching framework as proposed by Ranftl et al. [Ranftl 2012]. Their method is based on the continuous optical flow formulation of the stereo matching problem, which we now outline. Formally, we wish to estimate a disparity field $d(\boldsymbol{u})$ which indicates the disparity between a given point $\boldsymbol{u} = (u, v)$ in the left image and its correspondence in the right image $(u + d(\boldsymbol{u}), v)$. Initial methods adopted the brightness constancy assumption, i.e., that the intensity of these points in both images should be the same: $I_L(u, v) = I_R(u + d(u, v), v)$.

However, these intensity values are likely to be afflicted by lighting issues, so this correspondence relationship is often enforced by only enforcing the constancy on a transformed version of the images:

$$\mathcal{F}(I_L)(u, v) = \mathcal{F}(I_R)(u + d(u, v), v). \tag{8.1}$$

The transformations we apply are to preserve the geometry of the input views while mitigating lighting effects: they map an input pixel (and its neighborhood) onto a feature vector of predetermined length such as, e.g., to a different colorspace. Common choices for the image transformation $\mathcal{F}$ are histogram equalization, the census-transform [Zabih 1994, Hafner 2014], the correlation transform to simulate zero-mean normal cross-correlation [Drulea 2013], or a conversion to the CIEL*a*b* colorspace with removal of the luminosity component [Kennedy 2015, Bailer 2015]. It is also perfectly possible to combine several transforms by simply concatenating their representations [Xu 2012]. An alternative to these methods is to perform full color correction before performing the matching [Lu 2015, Ye 2017]. In the subsequent discussion we will omit the notation of the transformation in favor of brevity; the input images can have any number of data channels.

The discussion of the method from [Ranftl 2012] falls apart in four parts: a data fidelity term that expresses how well a disparity field hypothesis fits the observations, an occlusion estimation technique to only apply data fidelity on valid pixels, a regularization term to express the expected piecewise smoothness, and finally the optimization procedure to minimize this entire cost function.

8.1.1 Data fidelity term

We now formulate the cost function that we wish to optimize in order to find the best disparity field hypothesis, i.e. the one that minimizes the difference between the corresponding pixels:

$$E_d(\hat{d}) = \sum_{(u,v)\in\Omega} \|I_R(u + \hat{d}(u,v), v) - I_L(u,v)\|_2^2, \tag{8.2}$$

where the summation for this data term runs over all image pixels (u, v) in the left image that are unoccluded: the error term is only meaningful for those pixels in the left view that are also visible in the right view – we cannot say anything about pixels that are occluded in the right image, or fall outside of its image bounds. We take this into account, and limit the summation to these unoccluded pixels. Which pixels are occluded, and which are not, can be derived from the disparity field; we discuss this in more detail in the next section.

Given a disparity field estimate $\hat{d}(u, v)$, we can also try to reconstruct the right image based on the left as

$$\hat{I}_R(u + \hat{d}(u,v), v) = I_L(u,v). \tag{8.3}$$

This warped version of I_L should resemble the right view as much as possible; in an ideal scenario, it is exactly the same. Note that equation (8.2) implies that the warped image $\hat{I}_R$ should be similar to the original right image:

$$E_d(\hat{d}) = \sum_{(u,v)\in\Omega} \|I_R(u + \hat{d}(u,v), v) - \hat{I}_R(u + \hat{d}(u,v), v)\|_2^2. \tag{8.4}$$

The warping step is not straightforward, and is done by using a z-buffer as discussed in the next section. For each pixel in the left image, we fill its intensity value into the right image at the corresponding location. If two different pixels in the left image map to the same location, only the closest object gets to write its intensity value; the other pixel is occluded in the right image.

8.1.2 Occlusion estimation using a z-buffer

Given a disparity field $d(u, v)$ between the left image I_L and the right image I_R, we wish to figure out which pixels in the central image are visible in the other view, and which are occluded (or fall outside of the image boundaries). To do so, we adopt an approach similar to the z-buffer of rendering engines. The z-buffer of a rendering pipeline shows, for each pixel, what the distance to the closest object in sight of that pixel is. If the disparity field hypothesis implies that multiple pixels in the left image have their correspondence at the same location in the right image, then only the pixel closest to the camera is considered unoccluded. As the disparity

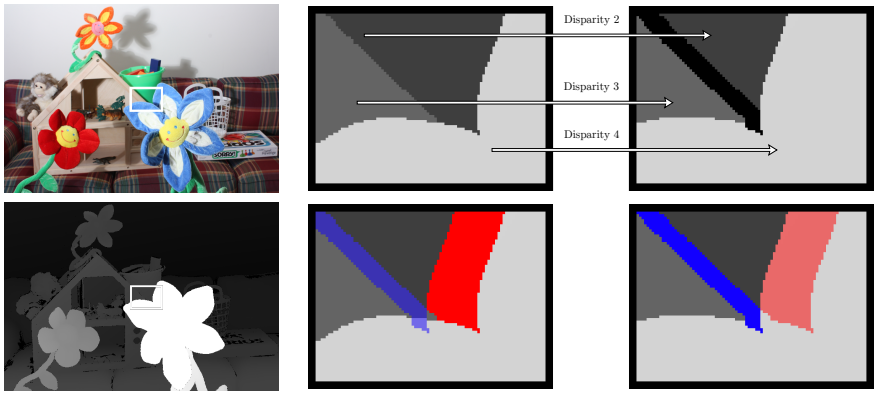


Figure 8.1: Occlusion detection for stereo matching using z-buffers. Left: input image (top) and ground truth depth map (bottom), with the crop highlighted. Right: the process of creating the zbuffer (by applying the disparity, top), and the resulting occlusion detections (bottom) illustrated on a small crop of the depth map. The blue pixels on the right are not visible at all in the original left image, and become unoccluded, from behind the light blue surface on the left image (although this is of little direct interest to us). More interestingly, the light red pixels in the right image get mapped on by both the flower in front and the background; the closest object wins and the original background pixels in the left image are marked as occluded.

is inversely proportional to the depth, the z-buffer will instead contain the largest disparity value rather than the lowest depth value. This is illustrated in figure 8.1.

Therefore, we need to figure out for a given pixel $\boldsymbol{u}_C = (u, v)$ which points are projected onto its correspondence's location $\boldsymbol{u}_R = (u + d(u, v), v)$. To do so, we define the z-buffer $\mathcal{Z}$ as the field that contains the disparity of the closest point that was projected onto each location in the right image:

$$\mathcal{Z}(u, v) = \max_{\{u' | u' + d(u', v) = u\}} d(u', v), \tag{8.5}$$

and we quantify the occlusion indicator field $o(u, v)$ in the central image as

$$o(u, v) = 1 - \exp\left(\frac{d(u, v) - \mathcal{Z}(u + d(u, v), v)}{\tau}\right), \tag{8.6}$$

where τ is a parameter we can use to indicate how much we trust the initial disparity estimate as in [Drulea 2013]. The closest point will always have occlusion score 0, and for $\tau \to 0^+$ all other points have an occlusion score of 1, indicating that we are sure of the input field $d(u, v)$. Correspondingly, the warped image $\hat{I}_R(u, v)$ can be reconstructed as

$$\hat{I}_R(u, v) = I_L(u', v), \text{ where } u' + d(u', v) = u \text{ and } \mathcal{Z}(u, v) = d(u', v). \tag{8.7}$$

In practical implementations, the discretization of the pixel grid requires us to perform interpolation of the various entities. In this case, we use the continuous scale of $o(u, v)$ to weigh the various contributing terms.

8.1.3 Smoothness term

So far, we have only local information for each pixel: there is no spatial reasoning of any kind, as every location in the disparity field is optimized independently from the others based on its own data fidelity term. We require spatial reasoning for two reasons: the disparity field should be locally coherent as it represents natural objects, and in the previous paragraph we have removed pixels from the error sum because of occlusions – for these, we need to propagate information from neighboring pixels. Therefore, we now introduce spatial regularization of the disparity field, the same one as adopted by Drulea et al. [Drulea 2013]. Based on the principle behind bilateral filtering, it allows depth discontinuities only in those areas in the image that also have intensity discontinuities: it assumes that any edges in the depth images also have to be present in the input images. Denoting the bilateral coefficient between pixels (u, v) and (u', v') as $b_{(u,v),(u',v')}$, the smoothness term $E_s(\hat{d})$ is defined as

$$E_s(\hat{d}) = \sum_{(u,v),(u',v')\in\Omega} b_{(u,v),(u',v')} \|\hat{d}(u, v) - \hat{d}(u', v')\|_1. \tag{8.8}$$

The bilateral coefficient is given by:

$$\begin{aligned} b_{(u,v),(u',v')} = & \exp\left(-\frac{(u-u')^2 + (v-v')^2}{2\sigma_d^2}\right) \\ & \exp\left(-\frac{\|I_L(u, v) - I_L(u', v')\|_2^2}{2\sigma_r^2}\right), \end{aligned} \tag{8.9}$$

which combines the first term, a spatial distance component, with an appearance component in the second term: we only penalize neighbors that differ in disparity if their color is similar and they are close to each other. Because in practice the σ_d that regulates the spatial component is rather low, the summation over (u', v') in equation (8.8) is only done over a local neighborhood $\mathcal{N}(u, v)$ for each (u, v).

8.1.4 Optimization of the cost function

We can now combine both cost terms into a single optimization problem:

$$\hat{d} = \underset{d}{\operatorname{argmin}}\, \lambda E_d(d) + E_s(d), \tag{8.10}$$

where λ is a weight term indicating the importance of data fidelity in relation to the smoothness of the resulting disparity field. An algebraic closed-form solution is intractable and unfortunately the bilateral smoothness term is also non-differentiable

because of the ℓ_1-norm, which inhibits direct gradient-based optimization techniques.

As in [Drulea 2013] we resolve this by adopting a primal-dual optimization technique. First, we introduce the forward linear operator K:

$$Kd(u,v,n) = d(\mathcal{N}_{(u,v)}(n)) - d(x,y), \tag{8.11}$$

where n is a linear index into the neighborhood $\mathcal{N}_{(u,v)}$. We now express the smoothness term from equation (8.9) as a scalar product by defining $F(\boldsymbol{q}) = \|\boldsymbol{b} \cdot \boldsymbol{y}\|_1$, where $\boldsymbol{b}$ is the correctly reshaped version of $b_{(u,v),(u',v')}$ so that both expressions are equivalent. Now, the smoothness term is given by $E_s(u) = F(Kd)$, and the optimization problem can be reformulated as a primal-dual problem without differentiability issues:

$$\hat{d} = \arg\min_{d} \max_{\boldsymbol{q}} \; E_d(u) + \langle Ku, \boldsymbol{q} \rangle - F^\star(\boldsymbol{q}), \tag{8.12}$$

where $\boldsymbol{q}$ is the dual variable representing the regularization penalty. In this expression, $\langle \cdot, \cdot \rangle$ is the inner product in $\mathbb{R}^{|\Omega| \times |\mathcal{N}|}$, and $F^\star$ is the Fenchel convex conjugate of F:

$$\begin{aligned} F^\star(q) &= \begin{cases} 0 & \text{if } q \in Q, \\ \infty & \text{elsewhere} \end{cases} \\ Q &= \left\{ q \in \mathbb{R}^{|\Omega| \times |\mathcal{N}|} | \forall (u,v) \in \Omega, n \in \mathcal{N}_{(u,v)} : \|q(u,v,n)\|_1 \leq b_{(u,v),n} \right\} \end{aligned} \tag{8.13}$$

We refer the interested reader to other works for more details on the application of primal-dual optimization in image processing [Chambolle 2011].

We optimize this problem by alternating between gradient descent (respectively gradient ascent) for the variables d and $\boldsymbol{q}$ (and after updating d, we recalculate the occlusion estimates). In order to make the data term differentiable, we use the first-order Taylor series approximation that is commonly used in the variational setting for stereo matching:

$$I_R(u + (d + \Delta d)(u,v), v) \approx I_R(u + d(u,v), v) + \Delta d(u,v) \frac{\partial}{\partial u} I_R(u + d(u,v), v). \tag{8.14}$$

Note that we need a non-zero horizontal derivative for a given location in order for the optimization to be meaningful there. This is called the aperture problem in stereo matching: we can only estimate disparities for areas of the image that have a pronounced horizontal derivative. For homogeneous areas the image intensity function is essentially constant, with the first and higher order derivatives zero or, worse, pure noise. For such ambiguous areas, as well as the previously discussed occlusions, we need the regularization term to propagate disparity estimates to ambiguous regions. As u and v are only sampled discretely, the relevant partial differentials are evaluated through differentiation.

We now optimize Δd in each gradient descent step rather than d. As we are iteratively optimizing in any case, because of the primal-dual nature, this first order approximation does not really impact convergence. However, as the Taylor approximation is only valid for small magnitudes of Δd, we need to take care not to take too large steps. Therefore, we also include proximal point terms to the optimization problem from equation (8.12), as in [Rockafellar 1976] (η and τ being tuneable stepsize parameters):

$$\hat{d} = \arg\min_{d} \max_{\boldsymbol{q}} \; E_d(u) + \langle Ku, \boldsymbol{q} \rangle - F^{\star}(\boldsymbol{q}) + \frac{1}{2\tau}\|d - d^{(i)}\|_2^2 - \frac{1}{2\eta}\|\boldsymbol{q} - \boldsymbol{q}^{(i)}\|_2^2. \tag{8.15}$$

In practice, we perform multiple gradient descent updates for each time we warp I_R to $\hat{I}_L$, for computational efficiency as well as to give the primal-dual problem time to converge. We will speak about the number of warps, and the number of optimization iterations within a warp in reference to this.

8.1.5 Hierarchical optimization

The final thing to discuss is the matter of hierarchical optimization, which already came up in the discussion of existing work. It is a long-standing practice in variational optical flow and stereo matching to perform the estimation in a hierarchical fashion: first estimating the correspondence fields on a low resolution, and then gradually refining it on the higher resolutions. This helps to mitigate the issue of local optima, which tends to be less pronounced on lower resolutions. As we discuss in the next section, the algorithm's complexity rises linearly with the number of pixels in the image; quadratically in terms of the resolution. Therefore, it makes sense to only spend the last few warps and iterations in the very highest resolution.

8.2 Stereo matching on high-resolution images

The input of high-resolution content to stereo matching is both a curse and a blessing. On the one hand, such images are able to capture fine details of the scene, and its textures, which offer valuable cues for the correspondence problem that are not present in lower resolutions. Yet on the flip side, higher resolution content means longer processing times and a possible aggravation of the ambiguous match problem (i.e., homogeneous areas and occlusions) because their absolute spatial extent increases. Whereas we can get around this issue with longer-range reasoning, the range required on high-resolution imagery grows unsustainably. This requires an increase in aggregation window size, or in the regularization window size; further increasing processing times beyond simply the quadratic rise caused by the increased pixel count.

This ambiguity of homogeneous areas is traditionally solved in a hierarchical way: first estimating the disparities on a lower-resolution version of the image, and then gradually refining on the higher-resolution versions up to the original resolution. In this way, the estimate from the lower scales yields a rough estimate for the higher resolution, where the optimization procedure can find the correct optimum if the initial estimate is close to it.

Still, every single pixel is processed, even though in the case of high-resolution images the disparities are likely to be very slowly changing and often many of the computations are redundant. Instead, we show in this section how to leverage the use of over-segmentation methods to arrive at a lower-resolution version of the input image that fulfills two important criteria:

- The sample points are still placed on a roughly uniform grid that we can interpret as a low-resolution image, and
- it is easy to recreate the disparity estimate at the input resolution from the disparity estimate on the over-segmentation grid.

When both points are met, we can simply apply the techniques discussed earlier on the subsampled grid which represents the over-segmentation, and afterwards upsample it to the original resolution. We show the reader the benefit of this approach, which not only speeds up processing but also offers increased robustness to noise on the input images.

8.2.1 Over-segmentation

We call *superpixels* the clusters of pixels that result from oversegmenting an image: groups of pixels that are similar, and hence don't contain major edges. We use the term *over-segmentation* here to indicate that these clusters do not correspond with physical objects on a one-to-one basis, contrary to traditional image segmentation [Zhang 2008]. Instead, one physical object is divided into many superpixels. The assumption is instead that this over-segmentation can be performed as a preprocessing step; the disparity of all pixels in a single superpixel is assumed to be constant or at least severely restricted. We now briefly discuss several over-segmentation methods, summarized in figure 8.2 and table 8.1.

Felzenszwalb's graph-based segmentation [Felzenszwalb 2004] is an efficient and popular segmentation algorithm, using a single scale parameter to influence the size of a segment. However, the actual size and number of segments varies greatly depending on local image content and it fails the first requirement mentioned earlier: it does not result in a roughly uniformly subsampled grid.

The normalized-cuts approach from Ren and Malik [Ren 2003] formulates the segmentation problem as a classification task between group labels; while results are good in general, it is a slow method. For the examples in figure 8.2, it was

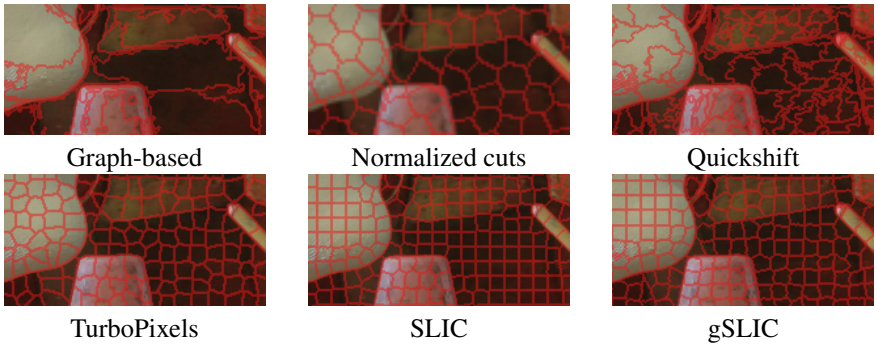


Figure 8.2: A visual overview of several over-segmentation methods.

Graph-based	Normalized cuts	Quickshift	TurboPixels	SLIC	gSLIC
1.428 s	7.45 min	15.18 s	71.7 s	0.503 s	45 ms

Table 8.1: Execution times for the various methods on an image of 1390×1110 *(Normalized Cuts was run on the half-resolution input). The used implementations were publicly available (C++ or MATLAB). gSLIC was implemented using Quasar [Goossens 2018].*

unable to process the full 1390×1110 image in an acceptable time — its results and running time are reported for half the original resolution.

TurboPixels [Levinshtein 2009] are based on geometric flow, providing superpixels that conform to image boundaries while keeping under-segmentation in check through a compactness constraint. It is much faster than normalized-cuts, but still lags behind the other methods in term of execution time.

Quickshift [Vedaldi 2008] is based on an approximation to kernel-based mean-shift. By processing pixels as points in a five-dimensional space (three color channels and two location coordinates), mean-shift in this five-dimensional space results in a segmentation of the original image. Similarly, SLIC [Achanta 2010] performs k-means clustering in this five-dimensional space. By controlling the seeds for the k-means algorithm, we can easily impose a desired average size on the superpixels, while a compactness parameter weighs the color subspace against the location subspace to provide a trade-off between clusters that are coherent in the spatial domain respectively in the color domain. One caveat with both Quickshift and SLIC is that these methods only encourage connectivity in the five-dimensional space. As we wish the superpixels to be connected in the location space, a post-processing step enforces this spatial connectivity.

Additionally, SLIC is very amenable to parallelization. As shown by Ren and Reid, the clustering algorithm can be adapted slightly to allow efficient execution on a GPU [Ren 2011]. Visually, the results are very similar to the original SLIC segmentation while executing several 10 to 20 times as fast. To do so, the im-

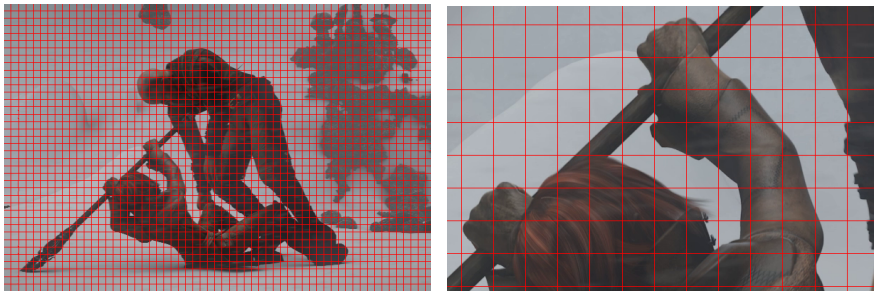


Figure 8.3: The initial clusters for gSLIC — a uniform grid. Left the full image, zoomed in on the right.

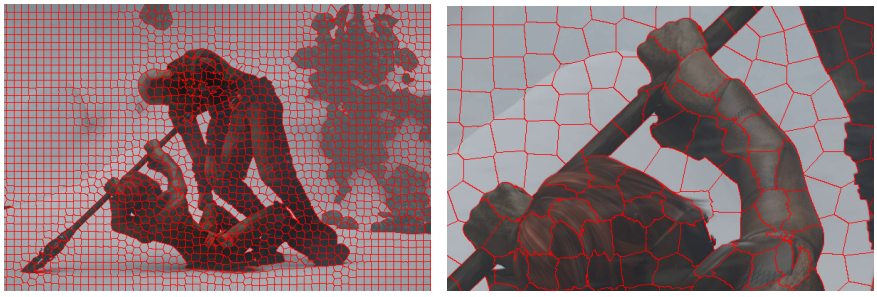


Figure 8.4: The final clusters for gSLIC. Left the full image, zoomed in on the right.

age is first divided into a regular grid of the desired size as in figure 8.3. During the optimization, there are two grids: the superpixel grid, which contains the location and color of each superpixel on this initial grid, and the pixel grid, which contains the superpixel each pixel currently belongs to. The superpixels' description is derived as the respective average, location or color, of all pixels belonging to it. After that, we iterate between updating the superpixel grid and the pixel grid: each pixel selects the most fitting superpixel in parallel as the closest one in the five-dimensional space, and after that we update the superpixel descriptions by recalculating the averages. Please refer to the work by Ren and Reid for a more in-depth discussion [Ren 2011]. An example of the gSLIC clustering with relatively large superpixels for better visualization is shown in figure 8.4. As mentioned above, the assumption behind our proposed approach is pixels in the same superpixel have nearly the same disparity. Therefore, we will use rather small superpixels, like those shown in figure 8.5.

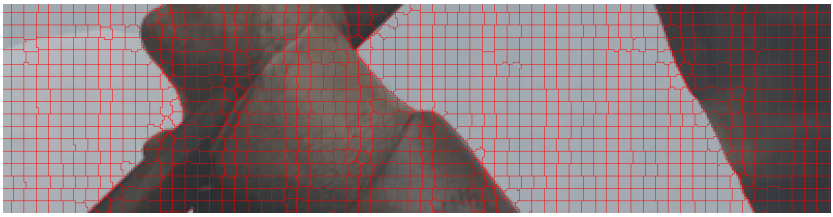


Figure 8.5: The much finer superpixel grid as it is used in over-segmentation-based estimation methods.

8.2.2 Over-segmentation for high-resolution image processing

The idea of the content-adaptive downscaling of images for efficient processing and storage proves to be a powerful tool in general [Kopf 2013, Heinrich 2013]. We will exploit this approach as a preprocessing step, before applying per-pixel stereo matching methods. Over-segmentation as a preprocessing step sees a widespread use in object segmentation and recognition, where it is often an initial step [Mori 2004, Mori 2005, Achanta 2012]. Initially, correspondence methods exploiting segmentation do so by semantically segmenting the image into objects [Nishigaki 2008, Van den Bergh 2012], with much larger segments as a result: this is semantic segmentation rather than over-segmentation. This is a difficult problem in and of itself, and the resulting segments are typically large and far from uniform. As a result, this requires specially adapted methods which can work on segmented images instead of pixels: graph-based techniques rather than techniques that work on regular grids. In contrast, our proposed approach does not require specifically adapted methods or a semantically correct segmentation. Another option is to use the over-segmentation to guide the estimation indirectly; rather than estimating one correspondence offset per segment, we can aggregate the errors over the segments (some form of adaptive aggregation window) [Chang 2013], or by regularizing pixels within the same segment [Gkamas 2011]. Neither of these techniques addresses the computational complexity for high-resolution images. In contrast, we apply the existing per-pixel techniques on the superpixel grid rather than on the full resolution input images for much more efficient processing while retaining the sharp edges of the image, as shown in figure 8.7. Compared to per-pixel processing on the full resolution, aside from the speed gain, we also benefit from a much larger effective spatial support of both the data term and the regularization term, which helps ignore local issues such as in figure 8.8, where the superpixel-based approach is able to correctly estimate the flow behind the blurred snow, which the pixel-based approach rather estimates the flow of the obstructing artefact (which is often undesirable).

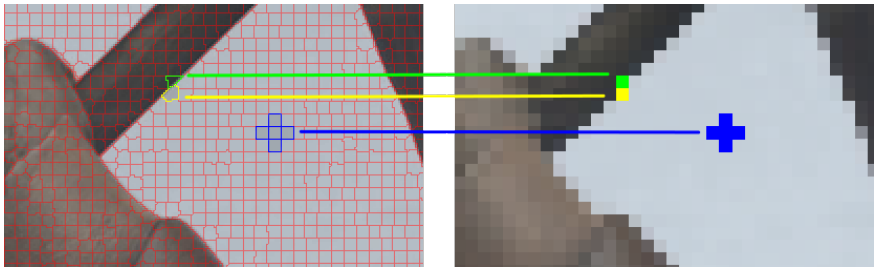


Figure 8.6: The clustering of the original image (left) and its mapping to the superpixel grid (right).

8.2.3 Variational processing on the superpixels

For stereo matching we create a superpixel grid for each input image, and process these using traditional per-pixel methods. This superpixel grid is a lower-resolution representation of the input image, as illustrated in figure 8.6. We refer to the earlier section 8.1 for the mathematical discussion of the variational stereo matching method we use here. The only adaptation we have to make is to take the precise location of superpixels into account for the regularization term, as they are no longer on a strictly uniform grid. Therefore, the bilateral total variation term from equation (8.9) is adapted to use the locations of the respective cluster centers (in the combined colour-space domain) rather than their coordinates on the superpixel grid.

To break up the monotony, the visualizations showed in this subsection will be for optical flow estimation. However, the variational optical flow method we have used [Drulea 2013] has also been applied with good effect to stereo matching with minor adaptations [Ranftl 2012, Donné 2015b, Donné 2017a] and the conclusions here are directly applicable to stereo matching, too. The difference is that in the optical flow problem, these correspondences don't necessarily lie on the same image row any more: the disparity field contains both horizontal and vertical offsets.

8.2.4 Impact of noise on the over-segmentation

We perform the full-resolution pixel-based as well as the over-segmentation-based approach on the test images from figure 8.9, which are full 4K resolution. In order to evaluate noise robustness at the same time, we apply Additive White Gaussian Noise (AWGN) with a standard deviation of 0% to 10% to the images. We find that gSLIC superpixels (and superpixels in general) are noise-robust: figure 8.10 shows how the clustering of the input image changes only slightly with respect to the increasing noise level, and the resulting superpixel grid cluster centers change even less.

Figure 8.7: Illustration of the over-segmentation-based methods for optical flow: the superpixel grids (top left) for both input images are processed by a pixel-based method, resulting in an optical flow estimate on the superpixel grids (top right). After resampling with the pixel label grid (bottom left), we arrive at an optical flow estimate for the original input image (bottom right).

8.2.5 Impact of superpixel size

We face a trade-off when selecting the superpixel size. First of all, there is obviously the speed up: working on the lower resolution superpixel grid speeds up the estimation noticeably. On the other hand, we must take care that we do not sacrifice accuracy for this benefit: larger superpixels allow the method to make larger-scale reasoning by increasing the effective size of its aggregation and regularization windows, but also assume all their pixels to have the same disparity.

Figure 8.11 shows the total execution time of both the existing dense pixel-based method on the original input image, as well as our approach for various superpixel sizes. We divide the execution time into three separate steps: applying gSLIC to acquire the superpixel grid, applying the dense method on the superpixel grid, and resampling to the original image resolution. We see that the total execution time is dominated by the variational optical flow (stereo matching) method used. At the same time, we have evaluated the difference in performance between a CPU and a GPU, leveraging the Quasar framework [Goossens 2018] that yields us both a CPU and GPU implementation from a single code base. Using superpixels of roughly 3×3, we achieve a speed-up factor of more than 40.

In order to evaluate the accuracy of the proposed approach, but abstract away the issue of homogeneous areas and occlusions, i.e. to evaluate how well the superpixel approach fulfills the cost function on the original resolution, we evaluate the accuracy as follows. First, we estimate the correspondence map (the disparity

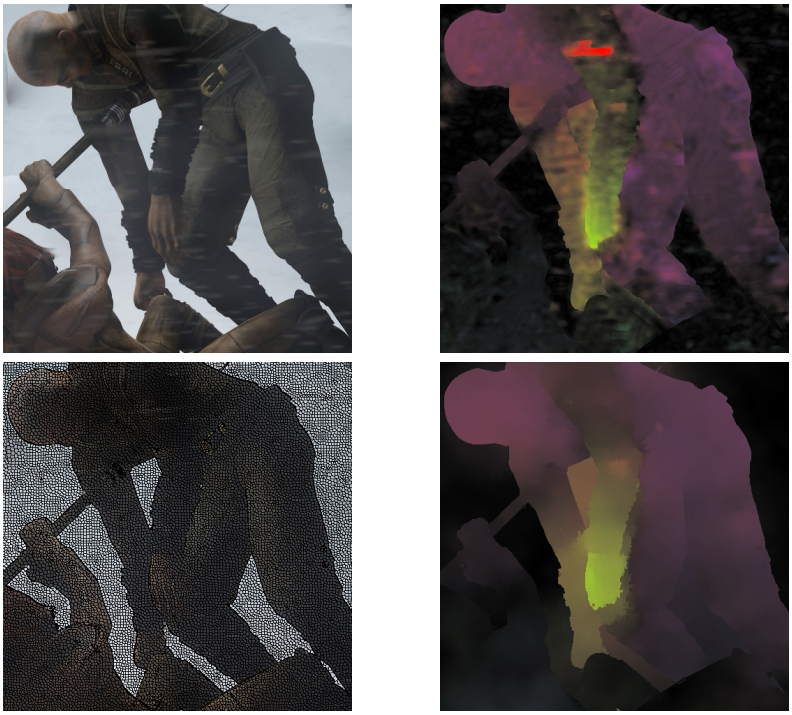


Figure 8.8: Top row: original input and estimated flow, bottom row: the superpixel segmentation and the corresponding flow estimate on the original resolution. Superpixel processing, asides from the speed benefit, also yields a larger effective spatial support for both the data term and the regularization term, which helps ignore local issues such as the snowflakes in these images (noticeably, the red spot on the arm).

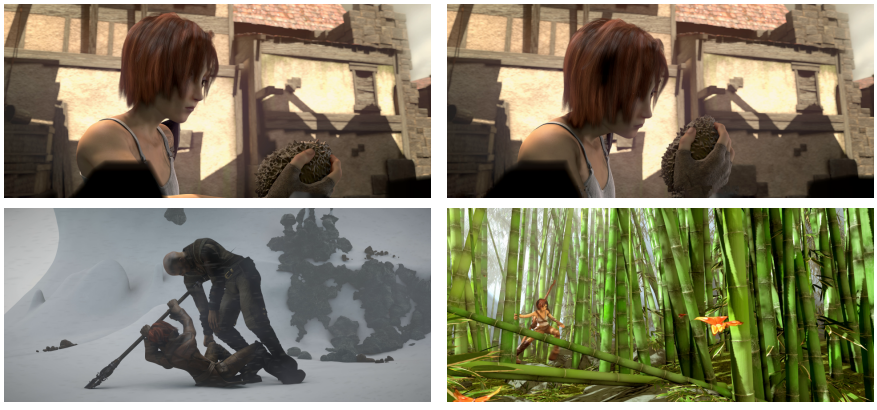


Figure 8.9: The test images we have used to evaluate the noise robustness and performance of the over-segmentation approach.

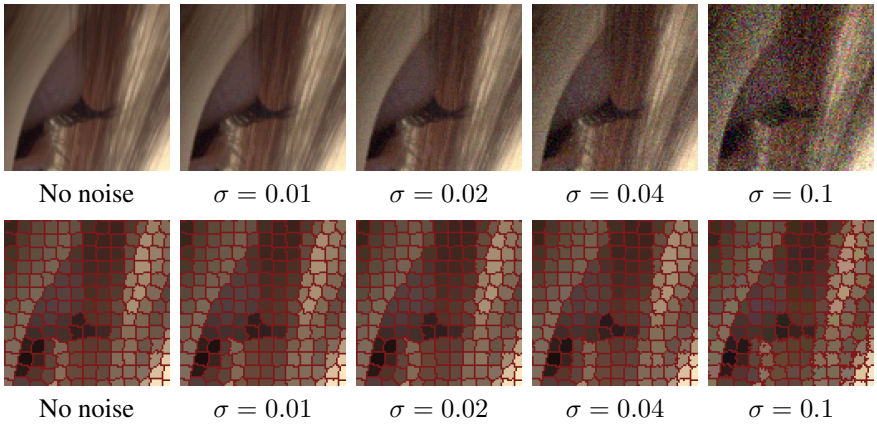


Figure 8.10: The result of SLIC over-segmentation on noisy images. Top: input images. Bottom: resulting SLIC over-segmentation. Note how the superpixel grid (the positions and colors of the superpixels) itself barely changes, even for high noise levels.

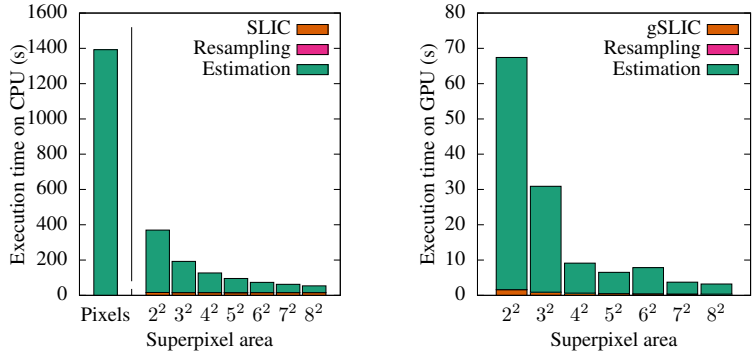


Figure 8.11: Timing data for the proposed method in function of superpixel size. The used CPU was an Intel Core i7-980X and the used GPU was an nVidia GeForce GTX 770. Due to memory restrictions, the per-pixel method was completely unable to be run on the GPU. The resampling step is negligible in all cases.

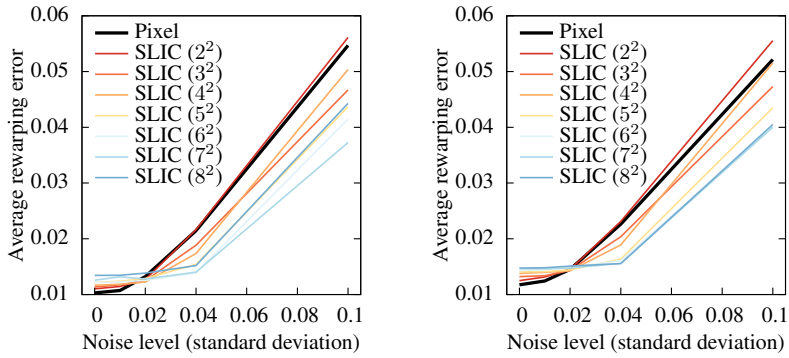


Figure 8.12: Noise impact for the top two test images shown in figure 8.9. Graphs for the other images are visually similar.

map in stereo matching or the optical flow) based on the noisy input images. Using this correspondence map, we warp (see section 8.3) the original input images onto each other; the resulting error is used to evaluate the quality of the estimate. In this way, we avoid both the noise and the ambiguous matching from influencing the error too much. The noise only indirectly matters, as it influences the estimation of the correspondence field. For the ambiguous matches, their characteristic aspect is that their matching cost is either nearly constant or effectively noise (otherwise they would not be ambiguous). Over large images, they are hence assumed not to influence the error beyond a constant offset. In this way, we shift the focus away from how well the specific per-pixel method handles occlusions and other ambiguities with a varying spatial range (due to the varying size of the superpixels), and towards the impact of the superpixels themselves.

The result is visible in figure 8.12. We see that the pixel-based method has a slight performance advantage over the over-segmentation method, but only in the absence of noise. As soon as the input images are perturbed by even a small amount of noise, the over-segmentation method benefits from the robustness of the underlying superpixels, outperforming the per-pixel method. As the superpixel size increases, so does the accuracy penalty in the noise-free scenario. However, the general trend is that the error rises less quickly under increasing noise levels.

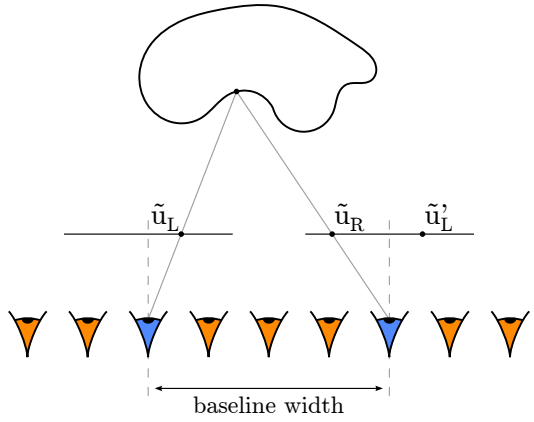


Figure 8.13: The multi-view stereo matching problem as we discuss it here, with all cameras positioned on the same baseline. The blue cameras are those from the traditional binocular stereo matching problem, but now we have more than two measurements along the baseline.

8.3 Extending binocular stereo matching to multi-view

Traditional stereo matching uses a binocular set-up: two cameras in a rectified configuration observe the scene, and from the disparity between corresponding pixels we can infer depth. However, such binocular techniques face problems with occlusions, pixels that are observed in one view but not the other, as we cannot directly estimate the disparity for those. So-called light-fields are captures made by multiple cameras on a baseline: the same set-up as binocular stereo matching, but with more than two cameras, as illustrated in figure 8.13. It is now unlikely for a pixel in one of the middle views to be occluded in all of the other views, so that at least some view pairs give us direct information from which we can estimate the disparity.

We have two specific applications in mind: first of all, imagine a camera moving along its scan-line direction. In that case the images are rectified from the start (we either ignore lens distortions or assume that they are corrected for because the camera was intrinsically calibrated prior to the capture). We call this the camera dolly set-up, and it is not far-fetched if we want to reconstruct a static scene; a simple home-made rig works well for this problem, as we illustrate in the experiments later on.

The other case is that of plenoptic cameras [Adelson 1992]. As introduced in chapter 2, a plenoptic camera is a camera that not only captures the intensity of the light in a scene, but also the direction that the light rays are traveling in space. In essence they consist of a set of micro-cameras on a regular, typically rectangular,

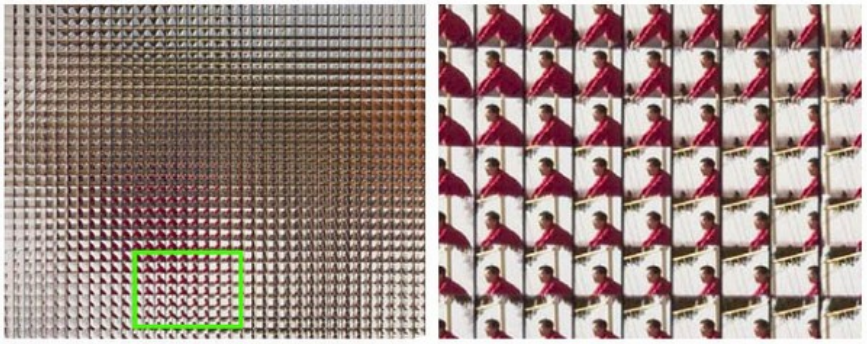


Figure 8.14: Extreme example of a plenoptic camera comprising many low-resolution cameras, courtesy of Adobe [Georgiev 2009].

grid. An extreme example, plenoptic cameras comprising many low-resolution cameras, can be seen in figure 8.14. As mentioned in chapter 2, there are two variants: unfocused and focused plenoptic cameras. The subsequent discussion assumes the first variant, in which each microlens in isolation can be considered a tiny camera system. It is clear that the rectification assumption is naturally fulfilled courtesy of the design: all microcameras share the same image plane, and the axes of the rectangular grid coincide with those of the image plane. Therefore, the method we discuss below is directly applicable to images from the same row of a light-field, but in section 8.3.1.4 we also discuss how to extend our proposed method to such a grid of 2D cameras.

We now show how we have leveraged the extensive literature from binocular stereo to the light-field setting; by cleverly extending a binocular method to allow for multiple images and correctly handling occlusions, we have designed a multi-view stereo matching algorithm that outperforms other light-field methods. To do so, we will continue building on the variational stereo matching approach as outlined in section 8.1. First of all, we extend the method to use multiple images on the same baseline in section 8.3.1, where we assume that the cameras' positions on the baseline are well known. Of course, mostly in the camera dolly setting, this is not always the case. Therefore, we also show how to adapt our extended stereo matching framework to estimate the camera locations themselves in section 8.3.2. The estimation of the camera locations is necessarily dependent on a 3D reconstruction of the scene in one form or another, and we motivate that using the same framework for the binocular depth estimation and subsequent camera location estimation for the other cameras on the baseline performs well. As the cost functions for both problems are intricately linked, ambiguous areas in the depth estimation case will not influence the camera location estimation much, as their contribution to the common cost function is flat, hence the ambiguity in the first case.

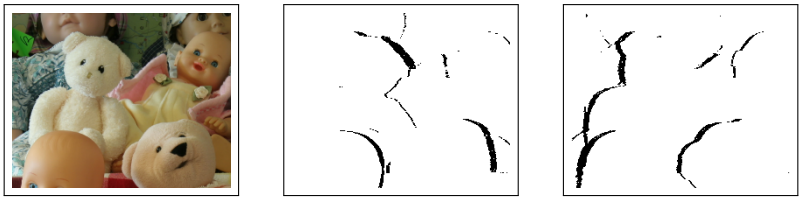


Figure 8.15: Pixels are unlikely to be unoccluded in all views, because the occlusion occurs in opposite directions in case of a central view. From left to right: the central view, the pixels not visible in the left view, and those not visible in the right view. The input is a detail from the dolls sequence of the Middlebury dataset [Scharstein 2003], and the occlusion maps were generated using the provided ground truth depth maps with the z-buffering technique mentioned in the text.

8.3.1 Multi-view stereo matching with known camera locations

The major challenge for binocular stereo matching is the handling of occlusions. While the binocular case cannot easily get around this and requires complex algorithms and techniques, we now take a look at an alternative: by using more than two views of the scene, but still restricted to the same baseline. When we are estimating the correspondence field for a central view I_C, pixels will typically not be occluded in all of the other views: pixels directly to the left of depth edges will be occluded in some views on the right side, and vice versa, but they will be visible in views at the other side of the central camera, as illustrated in figure 8.15. This allows us to mitigate the effect of occlusions by estimating the disparity of a given pixel based on only those views in which it is not occluded (see figure 8.15).

Such a stack of images is also called a light-field. Rather than viewing them one image at a time, we can also create light-field slices by concatenating the same scan-line from all images, as illustrated in figure 8.16. In traditional binocular stereo, such a view comprises only two rows and the depth of a point is related to the disparity offset between corresponding pixels. In the multi-view setting, the depth of a pixel is related to the angle of the line connecting all of its correspondences (assuming the cameras are distributed uniformly over the baseline, as is the case here). This has opened up the way for a new kind of stereo techniques that estimate these angles [Kim 2013, Heber 2013].

8.3.1.1 Multi-view disparity fields

Earlier, we have established that the disparity between a pixel in the left view and its correspondence in the right view is inversely proportional to the physical point's distance to the camera baseline as

$$z = \frac{fL}{u_L - u_R}, \tag{8.16}$$

Figure 8.16: Example of a lightfield slice, attained by concatenating the rows of a given scanline in all of the input images. Now, the depth of a point is related to the angle of the line connecting all its correspondences: the steeper (more vertical) this angle is, the further an object is to the camera baseline.

where f is the camera focal distance, L is the baseline width and the difference $u_L - u_R$ on the horizontal axis expresses the disparity between the point $\boldsymbol{u}_L$ and its correspondence $\boldsymbol{u}_R$. From the same relationship, we also see that the disparity for a given point (i.e. with fixed z) is related linearly to the camera baseline width. We call $d(u, v)$ the disparity field between a central view I_C and an arbitrarily chosen reference view I_R. We now create an axis on the baseline with its origin at the central camera's location and with the reference camera at unit distance from the central one. Now, given any other camera k at position θ_k along this axis, the linear relationship between disparity and baseline width lets us express the disparity field between the central camera and camera k as $d_k(u, v) = \theta_k d(u, v)$. Therefore, we have the same expression for each camera k as we had for the right view in binocular stereo matching:

$$I_C(u, v) = I_k(u + \theta_k d(u, v), v). \tag{8.17}$$

This expression is only valid for those pixels in the central view whose correspondences are visible in the k'th view. For binocular stereo this last caveat was a big problem: many pixels from the left image *were not* visible in the right view, and so we did not have salient information for all pixels. However, now it is unlikely for a pixel to be occluded in all other views; at least some of them will yield us some information. There are two issues left: estimating those occlusions, whose estimation based on a z-buffer we have previously discussed, and handling homogeneous regions. As the bilaterally guided regularization adopted in the binocular case handles homogeneous regions well, we will use the same regularization here.

8.3.1.2 Extending the cost function

We sum the data terms from equation (8.2) in the binocular case for all views (including the reference view) into a single dataterm. In order to have the same relative weighting between the data term and the smoothness term, regardless of

the number of images, we divide by the number of views:

$$\begin{aligned} E_d(\hat{d}) &= \frac{1}{K} \sum_k \sum_{(u,v)\in\Omega} \|I_C(u,v) - \hat{I}_k(u,v,\hat{d})\|_2^2 \\ &= \frac{1}{K} \sum_k \sum_{(u,v)\in\Omega} \|I_C(u,v) - I_k(u+\theta_k\hat{d}(u,v),v)\|_2^2. \end{aligned} \tag{8.18}$$

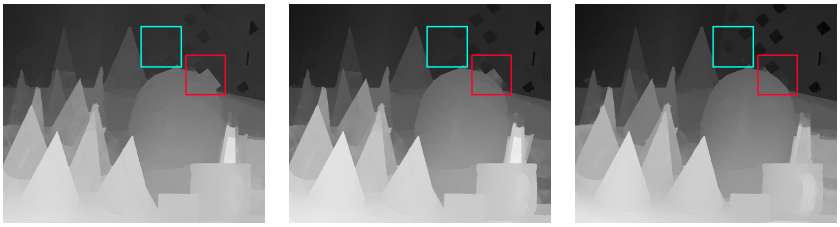
The regularization term $E_s(\hat{d})$ remains exactly the same as in the binocular case, as that was only based on the disparity field itself and the left view (which we now call the central view). Optimization is also handled in exactly the same way as outlined above for the binocular case, as our new data term is a linear function of familiar terms and the optimization technique is based on gradient descent. As a large part of the optimization complexity of the binocular problem stems from the proximal splitting algorithm (more specifically, from the dual variable for the regularization term), and we here have only expanded the data term, it turns out that the time complexity does not suffer as much as one might expect, for a small number of views. Estimating disparity with the above approach and changing no other parameters, we get a baseline time of $5\ 558$ms for the two-view case. Comparatively, the three-view case (twice as many side views) takes $7\ 360$ms, even though the data term requires twice as many calculations. After this, it increases linearly: for 11 views, ten times the number of side views, we require $22\ 073$ms.

8.3.1.3 Results

Now, we present our results to highlight the strength of our proposed approach, as well as the benefit of using multiple images. First, qualitative results show how the multiple images indeed alleviate the occlusion issue. Later on in this section, we show a qualitative comparison to several baselines.

In figure 8.17 we illustrate how more input images help the disparity estimation process. This is not only about the fine-grained accuracy (which is not easily visible to the human eye) but more about the big artifacts that are typically caused by occlusions, some of which were highlighted. A qualitative comparison with Kim et al., which estimates the angles in the lightfield [Kim 2013], shows that their method yields more noisy disparity maps because it does not have a regularization method built into it. However, due to the hierarchical optimization pipeline we adopt, we do sometimes miss very fine details as shown in figure 8.18.

In order to show results on (virtual) plenoptic camera images, we have estimated the disparity field for the mona dataset from Wanner et al. [Wanner 2013]. The result can be seen in figure 8.19. Note that while their method exploited the entire rectangular grid of cameras, we have only used the central line of cameras for our disparity estimate.



2 input views 3 input views 5 input views

Figure 8.17: Using multiple images, even a small number of them, strongly benefits the disparity field estimate. There are five images in the cones *dataset from [Scharstein 2003]. The left image shows the disparity field as estimated from images* 3 *and* 4*, while the middle image shows the result from images* 2 *through* 4*, and the rightmost image shows the result based on all available images. Notably, the disparity of the background object (the grating and the wall behind it) is increasingly successfully estimated the more input images we use.*

example input image our estimated disparity map

detail of the estimate by [Kim 2013] detail of the estimate by our method

Figure 8.18: Our disparity estimate for the couch dataset from [Kim 2013], using 10 input frames. Top row: one of the input images (left), and the disparity field we have estimated (right). Bottom row: a detail of the disparity field, as provided by Kim et al. [Kim 2013] (left), and as estimated using our proposed approach (right). In general our approach performs better, but it sometimes misses very fine details, such as the narrow metal bar.

Table 8.2: Quantitative comparison of our proposed multi-view stereo matching method with a planesweeping baseline and COLMAP [Schönberger 2016b, Schönberger 2016a]. The planesweeping baseline uses the basic correlation-based photometric error as outlined in the initial discussion on stereo matching, then blurs the cost volume with $\sigma = 2$ and then uses winner-takes-all. It used 64 depth hypotheses in a 50% bigger interval than the ground truth values, per image. We report mean absolute error, and three statistic stating how many of the estimates were more than 1, 2 or 4 pixels off the ground truth. We also report this for all pixels, and for only those the COLMAP method considers geometrically consistent (roughly speaking, the unoccluded ones).

all pixels	MAE (px)	sub1 (%)	sub2 (%)	sub4 (%)
planesweeping	1.0962	0.1968	0.1434	0.0903
COLMAP	1.0875	0.1902	0.1548	0.1029
our approach	**0.2861**	**0.0545**	**0.0368**	**0.0173**
unoccluded	MAE (px)	sub1 (%)	sub2 (%)	sub4 (%)
planesweeping	0.6009	0.1265	0.0789	0.0393
COLMAP	0.2879	0.0562	0.0421	0.0224
our approach	**0.2536**	**0.0436**	**0.0305**	**0.0161**

Finally, we compare quantitatively with COLMAP [Schönberger 2016b], a state-of-the-art multi-view stereopsis implementation. In general problems, it starts by first estimating the camera positions from sparse matches, and then uses the estimated positions to perform dense stereopsis. For the results in table 8.2, we pass it the ground truth camera positions and then estimate the dense depth maps. COLMAP additionally returns a mask for the estimates as to whether or not the depth estimates are geometrically consistent with those of the other views. From this, we perform two evaluations on the baselines: averaging over all known pixels (those present in the ground truth disparity) and averaging over those pixels that COLMAP considers geometrically consistent – broadly speaking, the unoccluded pixels. The datasets we use are those from Middlebury 2006 [Scharstein 2007], as they are the most recently release with multiple views per scene (7 views for each scene). We estimate the disparity field for the second view, which means that we don't have overly much information on occlusions: one side has one additional view, while the other side of the central camera has five views. We conclude that our proposed approach matches to slightly improves COLMAP's performance on the unoccluded pixels, and is actually able to more or less keep up this level of performance on the occluded pixels. COLMAP, in comparison, gets completely lost on occluded pixels. Taking occlusions into account lets our proposed method still achieve good estimates in those areas – taking such knowledge from binocular stereo matching techniques and applying them to multi-view stereopsis methods is a promising avenue of future research.

8.3.1.4 Extending the method to a 2D grid

It is straightforward to extend our proposed approach to a grid of cameras (parallel to the common virtual image plane) with known camera positions (θ_k, κ_k), such as is the case for plenoptic cameras. This requires only the reformulation of equation (8.17) as

$$I_C(u,v) = I_k(u + \theta_k d(u,v), v + \kappa_k d(u,v)), \tag{8.19}$$

and the reformulation of the Taylor series approximation from equation (8.14) as

$$\begin{aligned}\hat{I}_k(u,v,d+\Delta d) \approx \hat{I}_k(u,v,d) + \theta_k \Delta d(u,v) \frac{\partial}{\partial u}\hat{I}_k(u,v,d) \\ + \kappa_k \Delta d(u,v) \frac{\partial}{\partial v}\hat{I}_k(u,v,d).\end{aligned} \tag{8.20}$$

However, we have focused more on the camera dolly aspect than the plenoptic camera scenario due to its practicality and low cost, and leave this direction open for future research.

8.3.2 Multi-view stereo matching with unknown camera locations

Up until now we have assumed that the relative camera locations along the baseline are well known – typically because we know them to be uniform along the line.

Such may be the case for plenoptic cameras and carefully driven camera dollies, but even for those there are production errors or driver inaccuracies. On the other hand, for less controlled settings this assumption does not hold at all. Our testing rig consists of a clamp mounted on a rail track with a ruler attached. Optimistically, we can manually set the camera position accurately up to a fraction of a millimeter. Moreover, for faster captures we wish to simply slide the camera over the rail and reconstruct from that video sequence (assuming we slide slow enough not to have motion blur in the video sequence). The result is typically a non-uniform movement speed, as illustrated in figure 8.20.

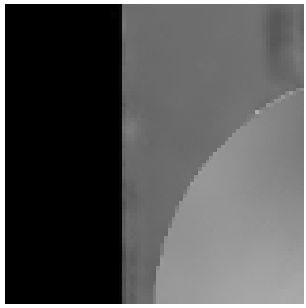
To address this issue, we estimate the precise camera locations based on the input images. The proposed approach fits within the stereo matching mindset: rather than using only a small subset of the input data (sparse image features) like the traditional structure-from-motion and SLAM techniques from chapter 5, we use the entire image as a dense data field for camera location estimation. By exploiting the stereo matching framework, we enjoy the benefit of dense matching: much more measurement points to base the estimation on, as well as our understanding of the cost function and its behavior with respect to occlusions and homogeneous areas.

The aperture problem highlighted by the Taylor approximation from equation (8.14) implies that the cost function contribution of homogeneous areas is

example input image

our estimated disparity map

detail of the estimate by [Wanner 2013]



detail of the estimate by our method

Figure 8.19: Depth estimation for the mona dataset [Wanner 2013]. Top row: one of the input images (left), and the disparity field we have estimated (right). Bottom row: a detail of the disparity field, as provided by Wanner et al. [Wanner 2013] (left), and as estimated using our proposed approach (right).

Figure 8.20: Example lightfield images from a camera moving along a dolly. Left: with a carefully controlled dolly, as prepared by Kim et al. [Kim 2013], the movement speed is constant and the depth of points is related to the angle of the line connecting their correspondences. Right: with manual movement of the camera along the rail, or another imprecise drive system, the camera does not move linearly at all, and it is far harder to visually gauge the depth of a given point.

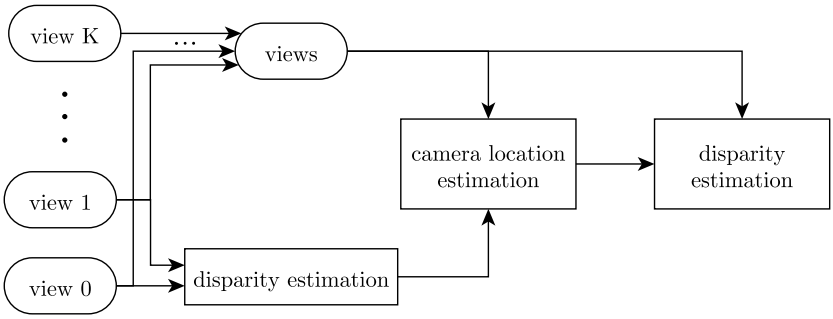


Figure 8.21: The full disparity estimation workflow. A rough disparity estimate from two views is used to estimate the camera locations for all views. Afterwards, those estimated camera locations and all input views can be used to estimate a more accurate disparity map for the chosen central view.

practically constant; such areas will not contribute meaningfully to the optimization problem of camera location estimation, even without any specific intervention from our end. We can hence simply ignore them. For occlusions, too, the cost function contribution is more or less noise. We could simply ignore it and assume the impact averages out to zero, but given the effectiveness with which existing works detect and handle occlusions we instead take these into account explicitly.

We introduce only one extra assumption: we assume that the camera movement is monotonous. The camera is assumed to only ever move in the same direction, albeit with varying (possibly occasionally zero) speed.

8.3.2.1 The entire workflow

The complete workflow for disparity estimation with multi-view stereo matching with unknown camera locations is shown in figure 8.21.

In the (largely hypothetical) scenario that the depth is accurately known, the camera location estimation can be performed directly on the input views and the ground truth disparity field. In the more likely case that the disparity field is not known in advance, one could envision an iterative process between optimizing the camera locations and refining the estimate of the disparity field – using the refined disparity estimate to improve the location estimates. However, our experiments have shown that there is little to no added value to this workflow. Presumably, this is because both optimizations use the same variational formulation and a similar cost function. Where the initial two-view disparity is wrong, this is because of ambiguities in the disparity problem formulation (e.g. due to homogeneous areas or occlusions). However, as we discussed in the previous section, such areas do not have a big impact on the cost function of the camera location either, by virtue

of that same ambiguity. Because the results of our initial experiments were not promising and we had a founded motivation as to why we do not expect this to yield notable improvements, we did not explore this avenue further.

8.3.2.2 The camera location cost function: the data term

We recap that the cost function for the disparity estimation was given by a smoothness term $E_s(\hat{d})$, in which we are not currently interested, and the data term

$$E_d(\hat{d}) = \frac{1}{K} \sum_k \sum_{(u,v)\in\Omega} \|I_C(u,v) - I_k(u + \theta_k \hat{d}(u,v), v)\|_2^2. \tag{8.21}$$

We now wish to estimate the camera positions θ_k, rather than the disparity field d. The result is the same data term, just with a different parameter:

$$E_d(\hat{\boldsymbol{\theta}}) = \frac{1}{K} \sum_k \sum_{(u,v)\in\Omega} \|I_C(u,v) - I_k(u + \theta_k \hat{d}(u,v), v)\|_2^2. \tag{8.22}$$

We again adopt the first order Taylor series approximation to linearize the function around the current estimate $\boldsymbol{\theta}$:

$$\begin{aligned} I_k(u + (\theta_k + \Delta\theta_k)d(u,v), v) \approx & I_k(u + \theta_k d(u,v), v) \\ & + \Delta\theta_k d(u,v) \frac{\partial}{\partial u} I_k(u + \theta_k d(u,v), v). \end{aligned} \tag{8.23}$$

We call $\hat{I}_k(u,v) = I_k(u + \theta_k d(u,v), v)$ the warped version of I_k with the current position estimate; similar to the warps from the disparity estimation, it is calculated by linear interpolation of I_k at positions $(u + \theta_k d(u,v), v)$.

Without any regularization or additional constraints on the camera positions, the resulting optimization problem separates into a set of relatively straightforward 1D optimization problems, as the position estimation of each camera is currently completely independent from the others. Sadly, there turn out to be many local optima for the separated per-camera objective functions (see figure 8.22) as well as a relatively small well of attraction near the correct minimum. However, when the current estimates are somewhat close to the correct optimum, the outlined approach converges to the correct optimum.

Our solution to avoid these local minima is two-fold. First of all, we will enforce the previously mentioned monotonicity of the camera locations: the order of the cameras along the line is the same as the order of the input images so that $\theta_k \leq \theta_{k+1}, \forall k$. Secondly, we work in a coarse-to-fine manner. In lower resolutions, the global minimum is much wider and local minima are rarer. By enforcing the monotonicity in the lower resolutions, for which the computations are not as expensive, we mitigate the effect of the local minima in higher resolutions because we now already have a rough estimate that is close to the global minimum. Consequently, we no longer enforce the monotonicity constraints on the higher levels as they are computationally expensive.

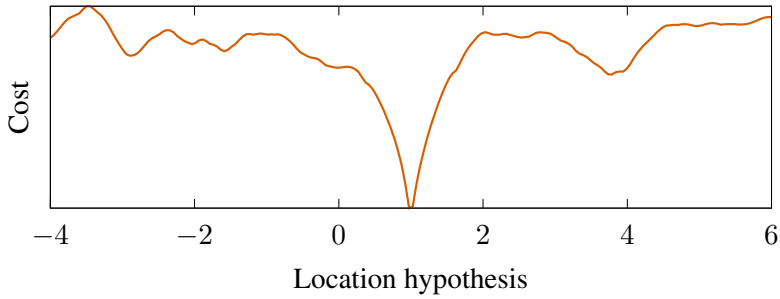


Figure 8.22: The cost function for one of the separated camera location estimation problems. Note the various local minima and the relatively narrow well of attraction around the correct optimum, 1.

8.3.2.3 Monotonicity constraints on the camera positions

After applying the linearization around the current estimate $\boldsymbol{\theta}$, the camera location estimation problem takes the form

$$\min_{\boldsymbol{\theta}} \sum_k \frac{1}{2}\beta_k \Delta\theta_k^2 - \alpha_k \Delta\theta_k, \tag{8.24}$$

where

$$\begin{aligned}\alpha_k &= \sum_{(u,v)\in\Omega} \langle I_C(u,v) - I_k(u+\theta_k d(u,v),v), d(u,v)\partial_u I_k(u+\theta_k d(u,v),v)\rangle,\\ \beta_k &= \sum_{(u,v)\in\Omega} d(u,v)^2 \left\|\partial_u I_k(u+\theta_k d(u,v),v)\right\|_2^2.\end{aligned} \tag{8.25}$$

Note that the values of α_k and β_k depend on the current location estimates θ_k and need to be recalculated for each warp. We take occlusions of the pixels into account by leveraging the same occlusion detection as in the disparity estimation framework (see section 8.1.2). Each pixel gets weight between 0 and 1 in the sums for α_k and β_k, based on whether or not it is occluded under the current location estimation.

Now, we introduce additional constraints: the new camera estimates still have to be monotonous (we will assume the previous ones were):

$$\begin{aligned}&\min_{\boldsymbol{\theta}} \sum_k \frac{1}{2}\beta_k \Delta\theta_k^2 - \alpha_k \Delta\theta_k,\\ &\text{where } \theta_{k+1} + \Delta\theta_{k+1} \geq \theta_k + \Delta\theta_k, \quad \forall k.\end{aligned} \tag{8.26}$$

This is a quadratic programming problem, for which many solvers exist. However, we keep in mind that this problem is only an approximation of the actual problem,

as it arises from a first order Taylor series approximation. Therefore, accurately solving the problem from equation (8.26) is of little relevance, and an approximate solution will do. We opt for an iterative approach based on Lagrangian duality. According to the theory of Lagrangian optimization, this problem is equivalent to minimizing the following Lagrangian w.r.t. $\boldsymbol{\theta}$ and $\boldsymbol{\lambda}$:

$$\begin{aligned}\Lambda(\boldsymbol{\theta}, \boldsymbol{\lambda}) = &\sum_{k=1}^{K} \frac{1}{2}\beta_k \Delta\theta_k^2 - \alpha_k \Delta\theta_k \\ &+ \sum_{k=1}^{K-1} \lambda_k \left(\Delta\theta_k - \Delta\theta_{k+1} - (\theta_k - \theta_{k+1})\right), \\ &\text{where } \lambda_k \geq 0.\end{aligned} \tag{8.27}$$

The resulting Lagrangian dual expresses the optimal value of $\Delta\theta_k$ for a given value of $\boldsymbol{\lambda}$:

$$\Delta\theta_k^\star = \frac{\alpha_k + \lambda_k - \lambda_{k-1}}{\beta_k}, \tag{8.28}$$

where $\lambda_0 = \lambda_K = 0$. This is the optimum of the unconstrained problem (α_k/β_k), corrected with a term to enforce the monotonicity. Substituting the Lagrangian dual from equation (8.28) into equation (8.27) results in another quadratic programming problem.

We use iterative Euclidean projection to enforce those positivity constraints, as the various variables are independent from each other. We calculate the optimal value for $\boldsymbol{\lambda}$ without the positivity constraints, project all λ_k on their feasible set (i.e. set negative values to zero). Iterative Euclidean projection was not straightforward in the original problem: the structure of the constraints was more complex. If two camera positions violated the monotonicity constraint, changing either may cause more violations with the cameras on the other side, so that the projection itself devolves into a complex problem. By introducing the auxiliary variables $\boldsymbol{\lambda}$ instead of the constraints, the projection is a simple operation.

Now we calculate the new values for $\boldsymbol{\theta}$ from equation (8.28) and continue with the next iteration. The gradient descent updates for $\boldsymbol{\lambda}$ are given by the solution of the system

$$\frac{1}{\beta_k}\lambda_{k-1} - \left(\frac{1}{\beta_k} + \frac{1}{\beta_{k+1}}\right)\lambda_k + \frac{1}{\beta_{k+1}}\lambda_{k+1} = \theta_{k+1} - \theta_k + \frac{\alpha_{k+1}}{\beta_{k+1}} - \frac{\alpha_k}{\beta_k}, \tag{8.29}$$

whose coefficient matrix is a symmetric tridiagonal matrix which can be efficiently inverted to solve the system [Mallik 2001, Kılıç 2008].

Our proposed approach is to perform several warps, and within each warp perform several optimization iterations of the primal-dual problem. We do not solve the primal-dual problem until convergence, as it is only an approximation of the problem in equation (8.26) anyway, but rather perform a predefined number of iterations.

8.3.2.4 The hierarchical approach

We use a hierarchical approach because the global minimum is more pronounced, and because the calculations on the lower resolutions are much faster. We downscale the input images by a factor two until the image size is smaller than 100×100. The camera locations are first estimated in the lowest resolution and iteratively refined on the higher resolution versions of the input. The discussed monotonicity constraints are only applied on the lower levels; higher levels simply use the closed form solution to equation (8.24). This speeds up the optimization as the computations in the lower resolutions are not as expensive. We can assume that for the higher resolutions the estimates are in the vicinity of the correct optimum and no longer need the monotonicity constraints. In our implementation, we only enforce the monotonicity on the lowest three resolutions: monotonicity constraints are only enforced up to $\sim 400 \times 400$, which was enough in all experiments we have performed.

It is interesting to note that variational stereo matching methods also use a hierarchical approach, albeit for a different reason. For them, the first order Taylor series approximation behaves better on lower resolutions. Additionally, the effective spatial extent of the regularization term is higher.

8.3.2.5 Overview of the location estimation

We now recap the workflow for estimating camera locations in a multi-view stereo matching scenario. First we, estimate the disparity field for the central view based on the central image and the reference image, which are said to be at respectively positions 0 and 1 along the camera position axis. Then, for each level of the resolution pyramid, starting with the lowest, we perform W warp iterations of the linearization around the current estimates of $\boldsymbol{\theta}$. Each such iteration is called a warp because it entails warping the images with the current estimate of the disparity field $\theta_k d(u, v)$. Per warp, we require the computation of the new values for $\boldsymbol{\lambda}$ and those of $\boldsymbol{\theta}$. We optimize $\boldsymbol{\lambda}$ with L iterations of gradient descent paired with the iterative Euclidean projection. The optimal values of $\boldsymbol{\theta}$ are then given in closed form by equation (8.28). The parameters for the reported results are: 10 warps at each resolution, and 10 updates of $\boldsymbol{\lambda}$ per warp before the new values of $\boldsymbol{\theta}$ are calculated.

The bottleneck of the algorithm is by far the calculation of α_k and β_k in each warp, because it entails a summation over all image pixels of the central view.

8.3.2.6 Results

The evaluation of our proposed method for estimating the locations is fourfold. First of all, we take a look at the monotonicity constraint and illustrate its effectiveness at guiding the estimate of camera positions far away from the central

camera. Secondly, we quantify the accuracy of our proposed method in terms of the estimated camera positions. We compare on both existing datasets from Kim et al. [Kim 2013] and new datasets we have captured in the context of [Donné 2017a]. In the third phase, we illustrate the method's effectiveness at estimating the camera parameters in a practical setting, plus how — and why — it can cope with typical stereo matching artifacts and errors in the input disparity map. Finally, we show visually how the proposed method transforms non-uniformly sampled lightfield slices into rectified versions containing only straight lines.

In our first experiment, we estimate the camera positions twice: once with and once without the monotonicity constraints. To this end, we use the statue and couch datasets [Kim 2013]. Kim et al. provide a large set of rectified views for both datasets, as well as high-quality estimated disparity maps. For a more challenging scenario, we use details of these views. This means that the image changes noticeably over the course of the camera movement along the baseline, as illustrated in figure 8.23: there is little overlap between the images at either end of the movement.

Another example is shown in figure 8.24. As the cameras are distributed uniformly (i.e. the camera moves at a constant speed), the position of the camera in terms of the frame number should be a straight line. The estimation procedure works well for cameras close to the central and reference camera, even without the monotonicity constraints, but cameras too far from the central one only yield nonsense results if we do not include the constraints.

In order to quantify the accuracy of the proposed method, we compare its performance with that of the VOODOO camera tracker [Thormählen 2012], a state of the art commercial camera tracking solution that has been used amongst others by Kim et al. [Kim 2013]. We perform this evaluation both on the Disney datasets [Kim 2013] and three additional datasets captured in the context of our own work: map, office and collection. These datasets were recorded with a camera mounted on a rail. The rail was annotated by a ruler, and we halted the camera for a while after every 5 mm (map and collection) or 1 cm (office). We automatically extracted the frames with the least inter-frame and visually inspected the extracted frames for correctness. We discuss three baselines:

- **VOODOO (projected)**: the VOODOO tracker tracks feature points through the sequence, and then performs unconstrained structure-from-motion estimation for the camera positions. We regress the resulting camera position estimates onto a single line to arrive at a scalar location for the cameras.

- **VOODOO (constrained)**: we use the same tracked features from the previous baseline, and pass them to a known-rotation solver [Hartley 2004, Kahl 2008b, Olsson 2007]. As we know that the rotation of the cameras is fixed and that the cameras only move along their x-axis, we only need

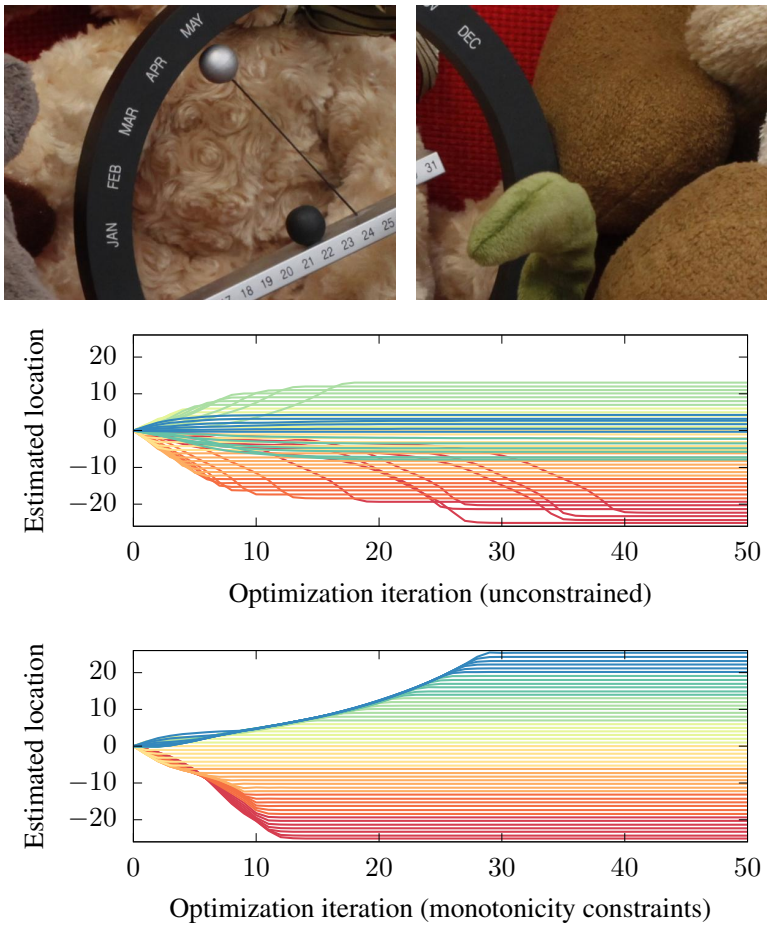


Figure 8.23: Importance of the monotonicity constraints. We estimate the camera positions of 25 cameras relative to the center two, which are said to be at position 0 and 1. In the top row we show the images from the outermost cameras: these don't overlap at all. In the middle, we show the resulting location estimation when no monotonicity constraints are used. In the bottom, we show the result when using the monotonicity constraints. Note that even for the locations that get estimated correctly in the unconstrained setting, convergence in terms of the number of iterations is faster when including the monotonicity constraints.

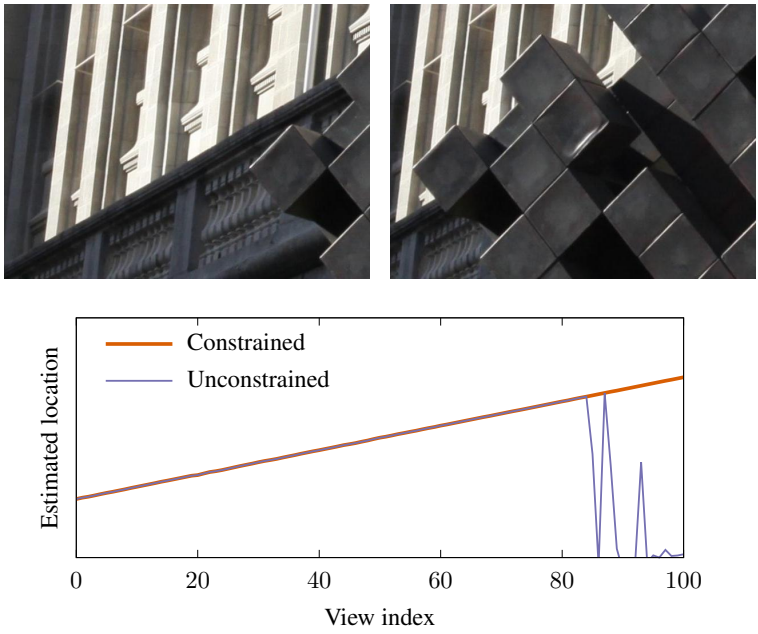


Figure 8.24: Importance of the monotonicity constraints, illustrated on a cropped version of the statue dataset [Kim 2013]. We estimate the camera positions of 50 cameras relative to cameras 15 *and* 25*, which are said to be at position 0 and 1. On the left, we illustrate both extreme camera positions. On the right, we have the estimated camera locations with (red) and without (blue) monotonicity constraints.*

to estimate a single value for each camera and a 3D location for each point. The result is a linear homogeneous system that is solved using singular value decomposition.

- **Our method (known depth)**: we also compare with the proposed method that has access to the ground truth disparity, if such information is available.

Finally, a note on the location reference systems. While our proposed approach assumes the central and reference cameras to be at positions 0 and 1 respectively, this is not the case for the VOODOO methods. They have an unknown reference system, and we need to somehow find the mapping from their reference system to that of our proposed method (and of the ground truth locations). We assume a best-case scenario, and exhaustively optimize this relationship for each evaluation separately to yield the lowest error possible given the VOODOO location estimates. Given the central camera C and the reference camera R, we can express

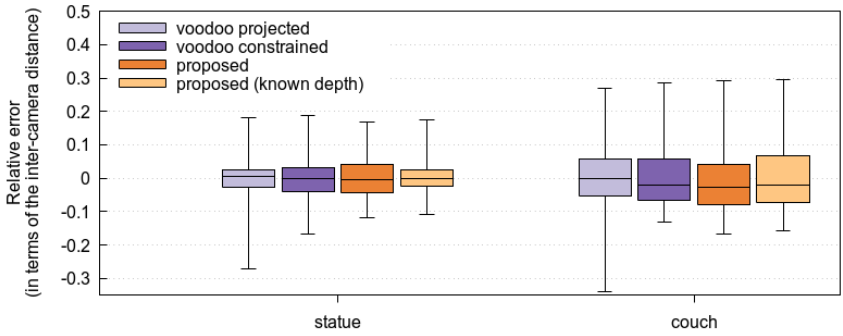


Figure 8.25: Comparison of the proposed camera location estimation and three baselines, on the cropped versions of the couch and statue datasets. The box plots show the average camera location error expressed as a fraction of the inter-camera distance (which is the same for all adjacent cameras) as well as the quantiles.

the mean squared error of the location estimates by our proposed approach as

$$\mathrm{MSE}_{\mathrm{proposed}}(\boldsymbol{\theta}) = \sum_k \left(\theta_k - \frac{k-C}{R-C}\right)^2, \tag{8.30}$$

but the estimates from the VOODOO tracking are not in the same reference system. So we optimize over the offset and scale of their reference system, independently for all experiments:

$$\mathrm{MSE}_{\mathrm{VOODOO}}(\boldsymbol{\theta}) = \min_{\mathrm{offset,scale}} \sum_k \left(\theta_k - \frac{k-\mathrm{offset}}{\mathrm{scale}}\right)^2, \tag{8.31}$$

As is visible in figure 8.25, the proposed method matches the VOODOO tracker for accuracy and performance, whether it uses the ground truth disparity map or whether it starts from the disparity map from binocular estimation.

We have additionally recorded a set of datasets (map, collection and office) with the previously outlined set-up of a rail-mounted camera and manually moving the camera to set locations. These datasets are illustrated in figure 8.26. In figure 8.27 it is visible that here we again improve or at the very least match the accuracy of the VOODOO tracking approach. The ground truth positions were obtained by reading the camera position off of the ruler attached to the camera rail track.

Finally, we test our proposed approach in a realistic set-up, without a controlled capture process: the camera was propelled manually, so that assumptions like constant speed or monotonous movement are not realistic. However, the camera's motion is still restricted to the rail, and it is fixed so that its scanlines are parallel to the movement direction. Consider now the input shown in figure 8.28,

Figure 8.26: The extreme camera positions for three datasets captured in our lab (from left to right: map, collection and office). The datasets were recorded using a rail-mounted camera (using a crank and chain mechanism). For the map and collection datasets, the distance to the scene is roughly 25-30cm, and the inter-camera distance is 5mm between all subsequent views. For the office dataset, the distance to the scene is roughly 2-5m, and the inter-camera distance is 1cm between all views. As our set-up has the camera moving vertically, the algorithms described here are applied on the transposed images so that the camera moves along the scanlines of these new images. All images are transposed again for any visualizations.

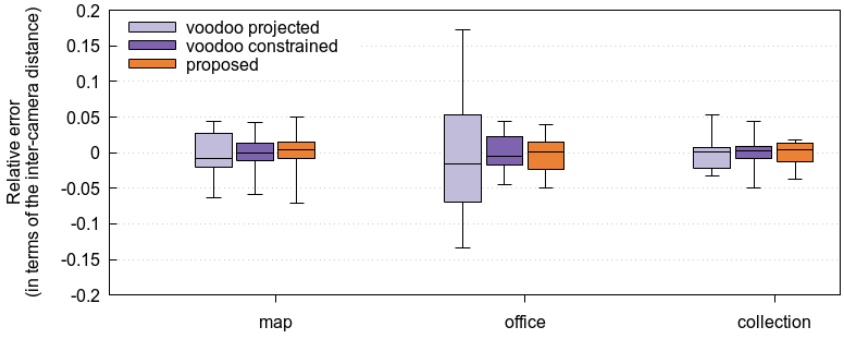


Figure 8.27: Comparison of the proposed camera location estimation and three baselines, on the three datasets recorded in our lab. The box plots show the average camera location error expressed as a fraction of the inter-camera distance (which is the same for all adjacent cameras) as well as the quantiles.

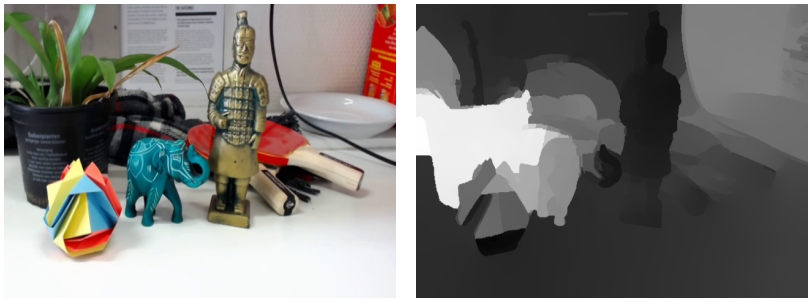
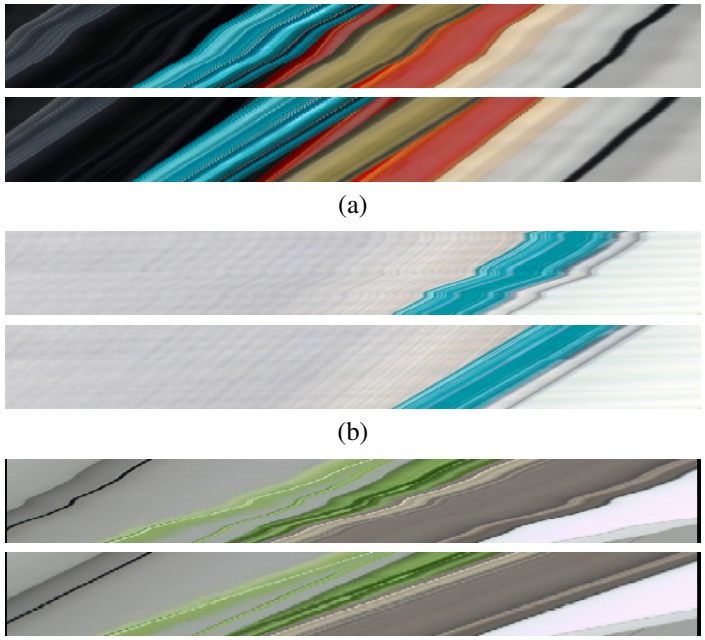


Figure 8.28: Example frame and initial two-view depth estimate for a manually recorded sequence.

and its two-view disparity estimate from binocular stereo matching. This is the input disparity we use for the camera location estimation. While the left portion of the image only offers an indisputably poor quality estimate, the correct right half still guides the camera location estimation correctly. As we have discussed before, our proposed method is not sensitive to regions that are ambiguous to the binocular stereo matching step. These bad areas are therefore not a big problem, assuming that a reasonable disparity estimate is available for a significant portion of the central image.

We visualize the estimated camera locations by using them to vertically resample the lightfield slices to uniform camera positions as illustrated in figure 8.29. If the camera positions of the lightfield slice are perfectly uniform, then it contains only straight lines. As illustrated in figure 8.29a, the faulty disparity estimate in the left part of the image does not impede a correct camera estimation, as the lightfield has been successfully resampled.

Qualitative results for the depth estimation using the estimated camera locations are shown in figure 8.30. On the left we show the disparity map estimate based on a faulty location hypothesis: uniform camera speed. The right image shows the disparity estimate after using our proposed approach to estimate the camera locations. One can see that the estimation exploiting multiple views (with the correct camera location) is able to avoid the issues in the left half of the input, as it now has views on either side of the central camera. Moreover, it is also visible that figure 8.30a contains errors that are not present in figure 8.30b, e.g. the depth of the elephant or near the edges of the statue. Correct estimates of the camera locations are paramount to a correct disparity estimate.

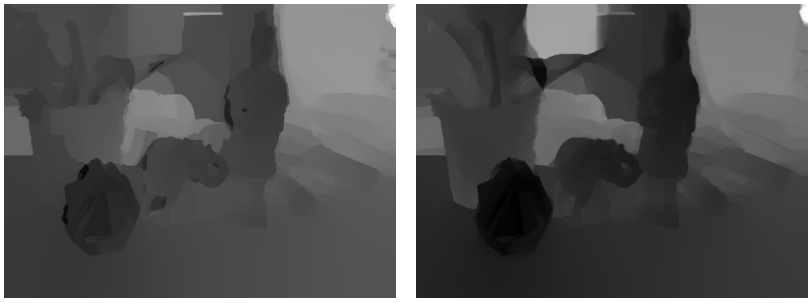


(a)

(b)

(c)

Figure 8.29: Input lightfield slices and their resampled versions after location estimation. In a lightfield slice from uniform camera positions, all lines are straight. When the camera speed was not constant, this is not the case and we can use the estimated camera locations to resample the lightfields. (a) shows the input and output lightfield slice for the dataset in figure 8.28, while (b) and (c) show two additional examples.



(a) (b)

Figure 8.30: The multiview estimation for the scene from figure 8.28. Both for wrong locations (assumed uniform, left) and after our proposed location estimation (right).

8.4 Contributions

In this chapter, we have discussed the estimation of a dense depth field for one central camera in a rectified set-up, the so-called stereo matching problem. After first introducing the existing body of work, we have first tackled the problem of very high-resolution inputs. After that, we propose an extension of the two-view rectified setting to mitigate one of the biggest issues with binocular stereo: occlusions. With known camera locations, this approach yielded good results compared to the existing work, and we have shown how to effectively estimate the camera locations in the case that they are not known. The work discussed in this chapter has led to two publications in international peer-reviewed conferences, and one international peer-reviewed journal paper:

- *Fast and robust variational optical flow for high-resolution images using SLIC superpixels*, Simon Donné, Jan Aelterman, Bart Goossens and Wilfried Philips. Advanced Concepts for Intelligent Vision Systems, ACIVS 2015 [Donné 2015a].

- *Variational multi-image stereo matching*, Simon Donné, Bart Goossens, Jan Aelterman and Wilfried Philips. International Conference on Image Processing, ICIP 2015 [Donné 2015b].

- *Line-constrained camera location estimation in multi-image stereo matching*, Simon Donné, Bart Goossens and Wilfried Philips. MDPI Sensors, 2017 [Donné 2017a].

We have shown how extending the binocular stereomatching framework to a line of cameras yields very good results. Similarly, we expect the extension from a linear array of cameras to a full 2D grid of cameras to yield an additional increase in performance still. On a separate note, we consider the biggest issue with stereomatching techniques to be their execution speed. We have partially addressed this by implementing the techniques on a GPU, and by efficiently subsampling the input images with superpixels. However, faster is always better, and future work is likely to be focused on execution speed as much as on accuracy.

9

Use-case: the BAHAMAS project

" It's the questions we can't answer that teach us the most. They teach us how to think. If you give a man an answer, all he gains is a little fact. But give him a question and he'll look for his own answers. "

— Patrick Rothfuss, *The Wise Man's Fear*

Throughout this dissertation we have discussed the entire pipeline for 3D reconstruction, from calibrating the various types of cameras to efficiently and accurately reconstructing scenes in 3D. In this chapter we discuss the application of those techniques in a practical problem: that of automatic measurements of maize plants. This was done in the context of a research project we undertook in cooperation with biotechnological researchers at the Plant Systems Biology (PSB) lab of the Flemish Institute for Biology (VIB): part of the BAHAMAS ICON project. The goal is to automatically quantify external properties of the plants (their phenotype) which will drastically lower the manual work for biotechnological researchers.

First, we start with an overview of the goal of the project: why the automatic measurements are required and the challenges they pose. Afterwards, we discuss each of the aspects of the project in sequence: the various cameras in the system and their calibration, the denoising of the inputs, the 3D reconstruction of the maize plants, and finally how to perform automatic measurements on the 3D models. Our contribution lies in the end-to-end design of the automatic modeling: while various steps of the process exploit the improvements from the rest of this doctoral work, here we bring everything together into a practical use-case.

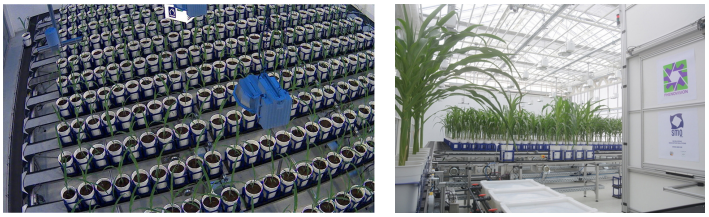


Figure 9.1: View of the conveyor belt system in the Phenovision system.

9.1 Automatically measuring phenotypes

An organism's genotype is decided by its genome, the set of genes it carries, while its phenotype is the name for all of its observable characteristics – influenced by both its genotype and its environment. Maize strains, genotypes, can differ in their response to specific abiotic stresses, such as drought or nutrient deficiency. The goal of Phenovision is to be able to quantify the impact of these external stresses on maize plants, charting which effect they have on the various genotypes of maize plants. To do this, the biotechnological researchers require large amounts of data and experiments. For example, typical required measurements are the amount of biomass in a plant (after drying), the speed of growth, ... While, e.g., accurately measuring the biomass of a plant after drying is a destructive process, the researchers wish to be able to model a plant's biomass through time, non-invasively and non-destructively. This opens up a whole avenue of new possibilities in research to give valuable insights into the effects of these stresses on the various maize strains.

To this end, the researchers at the Flemish institute for Biotechnology (VIB) in Ghent have designed and built Phenovision. Phenovision is an automated factory for high-throughput phenotyping of crops under greenhouse conditions. These crops are grown in pots that are transported on a conveyor belt system as in figure 9.1. Pots have unique identifiers, which make it possible to treat and image each plant separately. The plants are occasionally moved from the stationary growth area to the weighing and irrigation stations. Soil water and nutrient deficit conditions can thus be imposed on individual plants. Environmental parameters including air temperature, relative humidity and light intensity (photosynthetically active radiation), are continuously monitored in the greenhouse to direct the greenhouse heating, ventilation, humidification and lighting system.

A periodic visit to the imaging cabins enables the frequent measurements the researchers require. The two imaging cabins (see figure 9.2) are enclosed areas with controlled lighting conditions and a lift with a rotating platform. The first cabin contains a set of RGB cameras in a multi-view imaging set-up for the 3D reconstruction of plants in order to measure growth-related phenotypic traits.

Figure 9.2: The imaging cabins in the Phenovision system. On the left side the cabin for the RGB and LWIR cameras, and on the right the cabin for the hyperspectral cameras.

Aside from this, other physiology-related traits are measured by exploring a much larger stretch of the electromagnetic spectrum. A thermal infrared camera captures wavelengths in the range of 8-13 μm. In the second cabin, a state-of-the-art hyperspectral imaging system, consisting of a visible to near-infrared camera (VNIR, 400-1000 nm) as well as a short-wave infrared camera (SWIR, 1000-2500 nm), images the plants in a large swath of the spectrum. After camera calibration, we can relate locations on the 3D model to measurements in the infrared domain, expediting more involved analysis by biologists.

Throughout this entire process, we need to take into account that the measurement procedures should influence the plants as little as possible: for example, we avoid active 3D measurement techniques as discussed in chapter 5, as these introduce a non-trivial amount of light into the scene which might affect the plant growth. Furthermore, we need to keep in mind that PhenoVision produces many high-resolution images in quick succession, as well as an immense amount of information from the hyperspectral scanners. All of this data must be processed quickly enough not to result in a backlog of unfinished tasks. In practice, the system can process one plant pot every few minutes, so that our entire processing chain should not take longer than that. Having managed this, the processing of the captured data can be done locally on a single workstation.

9.2 Camera calibration

The first step is the calibration of the camera set-up. In order to be able to relate the measurements from one camera to another, we need the mutual relationships of the cameras. For the visual cameras we were able to use the traditional checkerboard pattern and the calibration method by Zhang et al. [Zhang 1999] as discussed in chapter 3. However, the hyperspectral cameras, which are linear-pushbroom cameras, highlighted an important issue with their respective state-of-the-art calibration method, and the thermal infrared camera posed a practical issue for the calibration object. We discuss first the calibration of the visual cameras, and then explain how to relate the other cameras to the same world reference system.

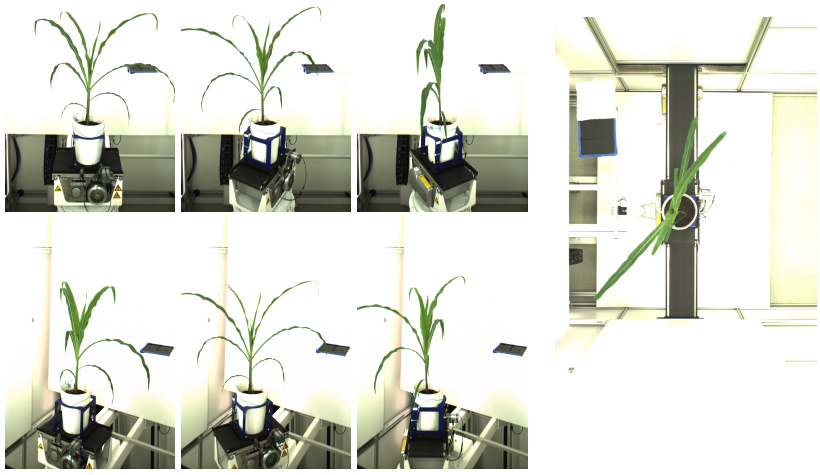


Figure 9.3: The visual-spectrum captures by the Phenovision system. Top left: the front-view camera, imaging three different rotations of the platform. Bottom left: the side-view camera, at roughly 30 degrees from the front-view camera, images the same rotations. Right: the top-view camera only images the initial position; further images would not yield more information from this camera. The images were cropped for better visualization, in reality the plants are contained completely within the images.

9.2.1 Calibration of the visual cameras

There are three physical cameras in the room: a front-view camera S0, a side-view camera S1 at a horizontal 30-degree angle from S1, and a topview camera T0. The central platform with the plant can rotate and the side-view cameras make three observations each, for positions 0, 60° and 120° of the rotating platform. The result is seven different views of the plant in the visual domain, as shown in figure 9.3.

We approached the calibration of the visual-spectrum cameras as follows. First, we calibrated the three physical cameras: we estimate their intrinsic and extrinsic parameters. For this we used the conveyor belt system's ability to rotate and elevate the mounted checkerboard. We detect the area of the image that is likely to contain a checkerboard, and then performed adaptive thresholding on that part of the image. The resulting black/white image is passed to the calibration toolbox. This process is illustrated in figure 9.4.

After detecting the checkerboards in all of the images, and so having obtained the requisite correspondences, we can calibrate these three physical perspective cameras using the techniques outlined in chapter 3.

However, at this point we still lack the calibration of the virtual cameras that

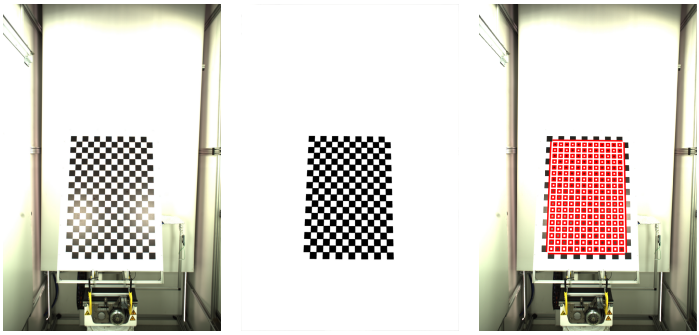


Figure 9.4: Detection of the checkerboard and subsequent calibration. Left: the input calibration image. Note the lighting issues because of the shiny surface of the board and the strong lights. Middle: the image after region-of-interest extraction and adaptive threshold. Right: the detected checkerboard overlaid on the original image.

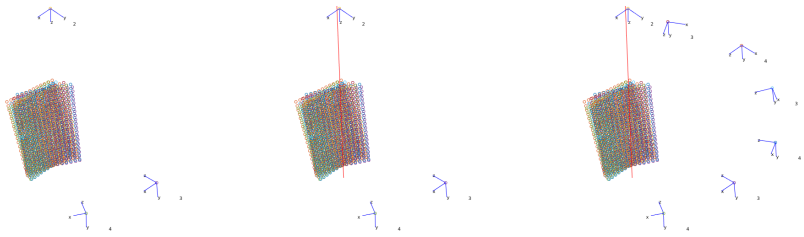


Figure 9.5: The calibration process of the visual-spectrum cameras. Left: first we calibrate the three physical cameras in the scene. Middle: next, we estimate the rotation axis of the robot. Right: finally, we use this rotation axis to express the four missing cameras in terms of the physical cameras calibrated earlier and the known rotations.

result from the platform rotation. As those are the same physical cameras, just at a different virtual location, we can copy the intrinsic matrices from the previous calibration and we only require their extrinsic matrices. We estimate the rotation axis from the point cloud of reconstructed 3D points from the checkerboards and use this rotation axis to create the four missing camera positions: the rotated versions of S0 and S1 under rotations of 60 and 120 degrees. This process is summarized visually in figure 9.5.

We are calibrating three physical cameras: the front-view camera F, the side-view camera S, and the top-view camera T. In the standard calibration process as discussed in chapter 3, we have estimated the intrinsic matrices K and extrinsic matrices $[R \mid \boldsymbol{t}]$ for each of these cameras. We have arbitrarily chosen the front-

view camera to be the provide the world coordinate system such that

$$[R_F \mid \boldsymbol{t}_F] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{9.1}$$

The calibration images were captured with a checkerboard that was mounted on the robot. For each of the calibration images, we know the rotation angle of the robot (with steps of 10 degrees). Furthermore, from the calibration procedure we also have the 3D locations of the checkerboard corners for each of these images. Now we collect all 3D points for each separate rotated position θ of the robot into a single point cloud P_θ. With an established geometrical approach called Procrustes analysis [Gower 1975], we estimate the transformations between each of these point clouds as a rotation around an unknown axis $\boldsymbol{u}$ with a known angle δ_θ and an offset $\boldsymbol{r}$. As these point clouds were expressed in the world coordinate system, we can then construct the extrinsic matrices for the virtual cameras of the front and side views under a given rotation θ of the robot:

$$\begin{aligned} R_{F,\theta} &= \cos(\theta) I_3 + \sin(\theta) \left[\boldsymbol{u}\right]_\times + (1 - \cos(\theta))(\boldsymbol{u} \otimes \boldsymbol{u}) \\ \boldsymbol{t}_{F,\theta} &= -R_{F,\theta}\boldsymbol{r} + \boldsymbol{r} \\ R_{S,\theta} &= R_{F,\theta} R_S \\ \boldsymbol{t}_{S,\theta} &= \boldsymbol{t}_S - R_{F,\theta}\boldsymbol{r} + \boldsymbol{r}, \end{aligned} \tag{9.2}$$

where $\left[\boldsymbol{u}\right]_\times$ is the cross product matrix of $\boldsymbol{u}$, $\otimes$ is the tensor product and I_3 is the 3×3 identity matrix.

9.2.2 Calibration of the LWIR camera

For the calibration of the LWIR camera in the first cabin, we faced a practical issue. In order to relate the LWIR camera to the visual-spectrum cameras, we require a calibration object that is easily visible in both modalities. We have briefly touched on this subject in chapter 4: because the thermal camera cannot resolve colors (there is barely any difference in surface temperature over the checkerboard), we cannot use the traditional checkerboard. With the help of the researchers at VIB, we instead used a perforated metal plate that was left out in the sun for a while before the captures. Because the surface has a light color, and the background is darker, we can easily spot the perforations in the visual spectrum. And because we left the plate out in the sun for a while, it is also noticeably warmer than the ambient temperature in the room. Its perforations are therefore easily visible in the long-wave infrared image too, as shown in figure 9.6. In order to detect the grate perforations, we manually annotate the four cornerpoints of the grate. The intermediate perforations are roughly located by simple uniform subsampling of those cornerpoints. These initial estimates are then refined by finding the local minimum

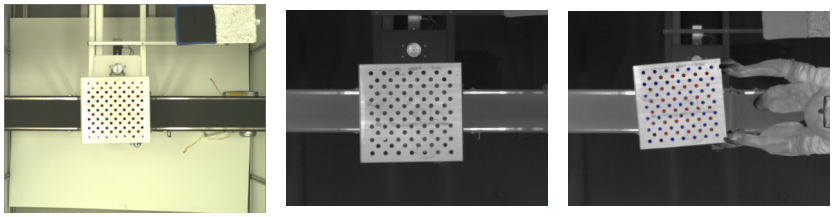


Figure 9.6: The calibration images for the thermal infrared camera. We move the metal grate both with the robot and manually, for more possible calibration positions (as the robot is not able to perform out-of-plane rotations). Left: top-view in the visual spectrum, with the grate fixed on the robot. Middle: top-view in the thermal spectrum, with the grate fixed on the robot. Right: top-view in the thermal spectrum, manually rotating the metal grate for more measurements.

of the gradient magnitude images. The detected positions are highlighted in figure 9.6. Our deep learning approach, MATE, for detecting checkerboard corners and other calibration points looked promising to apply to this scenario: with a relatively low number of manually annotated ground truth images, we could train the network to locate the grate perforations automatically. However, as the calibration only needs to be done once, we have used those manual annotations directly for the calibration procedure. If the calibration process needs to be done in the future, those images could be used to train a new version of MATE that can automatically detect the corners.

After finding the locations of the grate perforations, we can again use the correspondences between the visual-spectrum and long-wave infrared images to calibrate the LWIR camera with respect to the top-view camera. As we know the position of the top-view camera in our world reference system from the previous section, transitivity yields us the position and orientation of the LWIR camera, too.

9.2.3 Calibration of the hyperspectral cameras

For the calibration of the hyperspectral cameras (visual to near-visual infrared, and short-wave infrared), we were able to successfully use a traditional checkerboard object and corresponding detection techniques (such as our own proposed approach, MATE, from chapter 4). Because the spectra are so close to the visual spectrum, the black-and-white squares of the checkerboard are distinctly on the calibration images, as in figure 9.7.

However, these cameras are fundamentally different from the perspective camera model of the visual-spectrum and LWIR cameras. Instead, they are linear pushbroom cameras as introduced in chapter 2: perspective cameras in one coordinate but orthogonal in the other, because they sweep a linear sensor array over the scene.

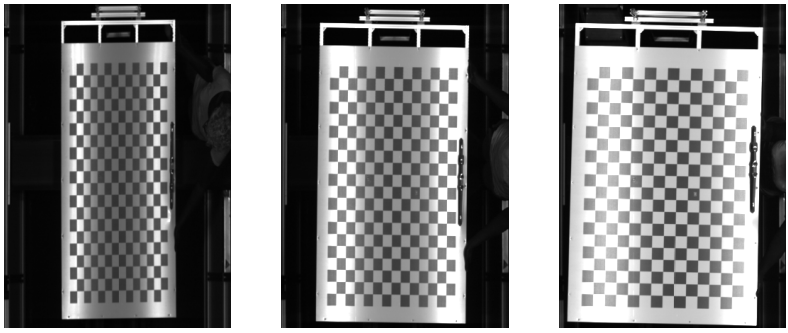


Figure 9.7: Calibration images for the hyperspectral cameras. We extracted one wavelength from the SWIR image to illustrate the peculiar nature of a linear pushbroom camera: from left to right, the checkerboard is brought closer to the camera. For a traditional perspective camera, the checkerboard would appear larger in every direction. However, for the linear pushbroom camera, the checkerboard only appears larger in the horizontal direction (the perspective coordinate axis), while the vertical size is unaffected (the orthogonal coordinate axis).

Unfortunately, the state of the art for calibrating these types of cameras at the time was not able to cope with some "degenerate" checkerboard positions; most notably, checkerboards parallel to the image plane (i.e. the plane defined by the sensor array and the movement direction), as discussed in chapter 3. The method by Drareni et al. cannot handle certain elements of the camera rotation matrix being zero [Draréni 2011], and yields only nonsense results in those case. Unfortunately, in our set-up, the planes were always nearly parallel to the camera due to the restrictions of the robot and the capture system, and those relevant rotation matrix entries were always nearly zero. Accordingly, we have designed and published a more robust approach for linear pushbroom camera calibration with checkerboards, which solves this issue and allows us to relate these hyperspectral cameras to the visual-spectrum cameras and the world coordinate system. At this point we have therefore calibrated all of the cameras present in the Phenovision set-up.

9.3 Denoising the hyperspectral images

Hyperspectral images are very noisy, especially in the low-response wavelengths (the edges of the measured wavelength range). Additionally, they contain a significant amount of structured noise, as also shown in figure 9.8. In order to help the biologists analysing these images, we denoise them to suppress these issues.

For the denoising of these images, we used the multi-component sparsity technique as proposed by Liao et al. [Liao 2013]. We refer to that publication for the

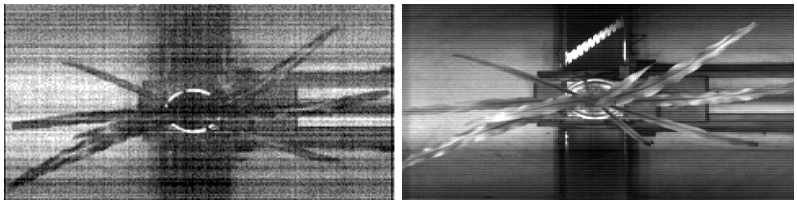


Figure 9.8: The high amount of noise in the hyperspectral captures. Left: for the VNIR images, the noise is most obvious in the low-response wavelengths. Right: for the SWIR images, there is structured noise even in the high-response wavelengths. The contrast of these images was enhanced for printing.

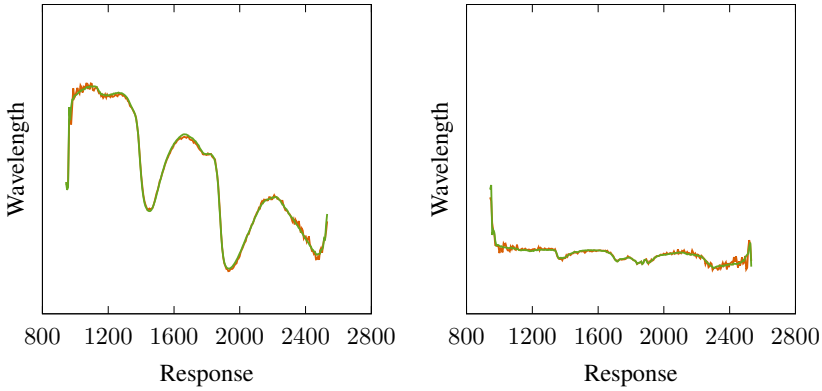


Figure 9.9: Some visual results after applying the denoising technique from [Liao 2013]. In red we show the input response, and green shows the denoised values. The biggest impact here is visible in either end of the graph: the low-response channels. Left: a plant pixel. Right: a background pixel. The y-axes have the same scale.

theoretical details, only mentioning that it enforces coherency both in the spatial and in the spectral domain. In this way, it is able to cope both with the structured noise and the low signal-to-noise ratio in the low-response channels — some results are shown in figure 9.9 and figure 9.10. Our contribution in this has been that we have implemented the method efficiently on the GPU. This has resulted in a speed-up of factor 20: whereas the original implementation took on the order of several minutes per scan for denoising, it is now a matter of seconds to denoise the hyperspectral scans of a single maize plant on a consumer GPU. This is important when Phenovision is running at full capacity: roughly speaking it could image one plant every couple of minutes, so in order not to create a backlog of data, we should take care that our whole processing chain takes less than that.

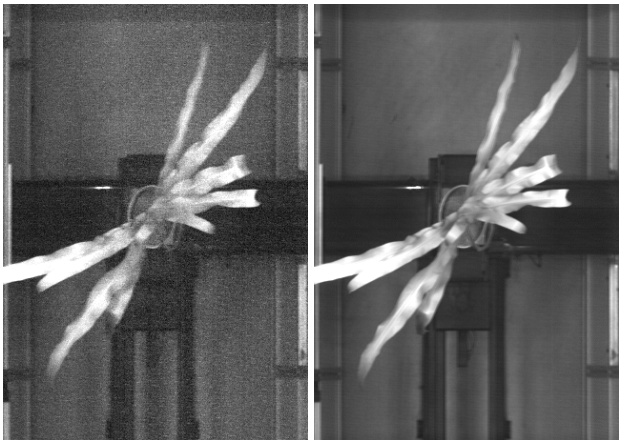


Figure 9.10: Some visual results after applying the denoising technique from [Liao 2013]. We show one of the low-response slices of the SWIR cube, where the low signal-to-noise ratio is especially visible. After the procedure, the noise is drastically diminished without loss of details or sharp features in the image.

9.4 3D Reconstruction of the maize plants

Recall that we wish to perform automatic measurements with regards to the growth speed of the plants, the number of leaves and their length, etc. To this end we require 3D information about the plant, the extraction of which we discuss in this section. We use voxel carving, also known as shape-from-silhouettes, in order to get a 3D reconstruction of the plant. As the name implies and as mentioned in chapter 5, shape-from-silhouettes requires the silhouettes for all of the input images. First, we discuss how to acquire these silhouettes and how to perform the subsequent voxel carving. After the voxel carving, we model the plant as a stem spawning a set of offshoots to arrive at a parametric modeling of the plant. All this together allows us to extract the automatic measurements as requested by the biotechnological researchers.

9.4.1 Segmentation of the input images

In order to segment the input maize plant images, we have opted for a machine learning approach. Even though the imaging parameters (lighting, camera shutter time, etc.) are controlled by Phenovision and generally speaking very good, manually defining the segmentation rules is cumbersome and finicky. Due to the fact that the input images are of such high quality, we do not need a complex or deep machine to segment the images. Convolutional neural networks of only a handful of layers have already proven very proficient at various forms

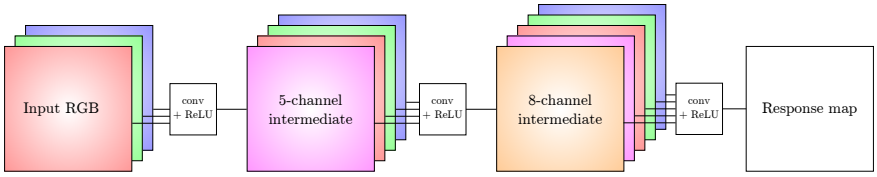


Figure 9.11: The architecture for our segmentation CNN for the maize plant images.

of pixel-wise processing: from super-resolution [Dong 2014, Dong 2016] to optical flow [Ranjan 2016, Dosovitskiy 2015, Ilg 2016]. It has already been used for segmentation in both MRI images [Powell 2008] and electron-microscopy images [Roels 2016]. Therefore, we also expect it to yield good results in our relatively straightforward problem.

Our proposed architecture is shown in figure 9.11. It is a convolutional neural network (CNN) that transforms the input RGB image into a single-channel response map, which can be simply thresholded to yield the segmentation. After every intermediate convolution, there is a ReLU activation layer to introduce non-linearities. You can see some of the resulting segmented images in figure 9.12. An initial version of the neural network was presented at the BENELEARN conference [Donné 2016e], but we have since improved on the network design, by introducing the skip-connections: the input to a given layer is the output of all previous layers, rather than only the output of the last layer. Because of the strong visual cue we have that even the original RGB channels contain strong evidence for the segmentation, we want to make sure this information is still available to the last layer of the network, hence the skip-connections. This principle was since introduced formally by Huang et al. [Huang 2017] as densely connected convolutional networks, and has proven to be very effective at a wide range of tasks. The network was trained in exactly the same way as the checkerboard detection network in section 4.2.2.4, please refer to that section for details.

9.4.2 Voxel carving

For the reconstruction of the maize plants we use the voxel carving approach we introduced in chapter 5. Alternatively, it is also called shape-from-silhouettes or visual hull [Laurentini 1994]. In voxel carving, we estimate the occupancy of each voxel in a dense $D \times H \times W$ grid. We project each voxel (d, h, w) onto each of the input images. It is only considered occupied if it is projected onto a foreground pixel for all of the images: as soon as one camera sees that location to be free, it is considered empty. Formally, we have:

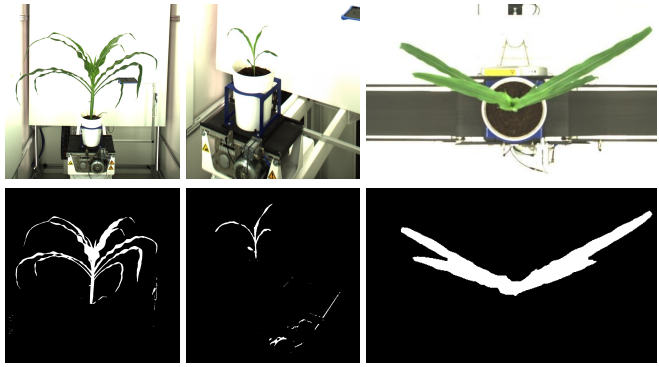


Figure 9.12: Visual results of the maize plant segmentation by our trained neural network. Note how even very small details that are hard to distinguish visually are also segmented correctly. The artifacts that remain from the background are neatly removed from the 3D reconstruction because they remain stationary, i.e. in the virtual reference systems, they are no longer in the same position. The images were cropped to show the relevant portions.

$$\mathcal{O}(d,h,w) = \min_c I\left(u_c, v_c\right) \text{ with}$$
$$\begin{bmatrix} u_c \\ v_c \\ 1 \end{bmatrix} \propto K_c \left[R_c | \boldsymbol{t}_c\right] \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = K_c \left[R_c | \boldsymbol{t}_c\right] \begin{bmatrix} x_0 + (d+1/2)s_x \\ y_0 + (h+1/2)s_y \\ z_0 + (w+1/2)s_z \\ 1 \end{bmatrix}, \tag{9.3}$$

where $[x_0, y_0, z_0]^{\mathrm{T}}$ is the position of the top left of the grid expressed in world coordinates, $[s_x, s_y, s_z]^{\mathrm{T}}$ contains the size of the voxels, and the half-voxel offset is used to project the voxel centers rather than their top-left corner. Off-grid sample points of $I(u, v)$ are evaluated using linear interpolation.

We reconstruct the maize plants at a resolution of $256 \times 256 \times 256$. While the consumer graphics card we were implementing on was able to cope with higher resolutions ($512 \times 512 \times 512$ was still very feasible), the lower grid resolution is able to represent all relevant details. Increasing it only served to increase the number of computations required. This is illustrated in figure 9.13.

9.4.3 3D Reconstruction on a GPU

It is interesting to discuss the implementation of shape-from-silhouettes, more specifically its implementation on GPU. An intuitive view of equation (9.3) is as follows. Given a set of silhouettes, we carve away at the voxel grid: initially the grid is fully occupied, and for each input image we cut away those voxels that are seen as empty by that camera. This is illustrated for two dimensions in figure 9.14, but the procedure is exactly the same in 3D, which we visualize in figure 9.15.

Figure 9.13: Result of the voxel carving of one single maize plant, through time (captures one week apart).

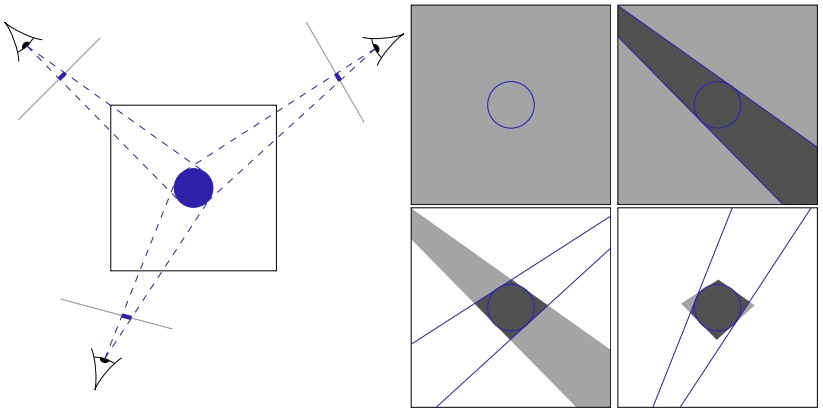


Figure 9.14: Illustration of voxel carving in 2D. Left: three cameras are observing a scene within a known bounding box, containing a square. Right: Voxel carving, image-per-image. Starting with a fully occupied voxel grid, we iteratively carve away regions that are seen as empty by the subsequent cameras: we intersect the current set of occupied voxels by those that are observed by the current camera as being occupied.

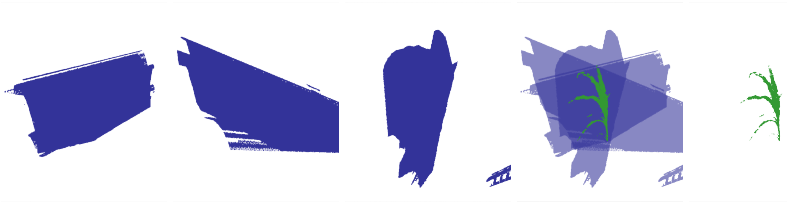


Figure 9.15: Voxel carving in 3D works in exactly the same way: a series of cameras is used to cull away empty voxels, until only occupied voxels remain. The resulting reconstruction is the intersection of all silhouette cones, based on the known camera parameters.

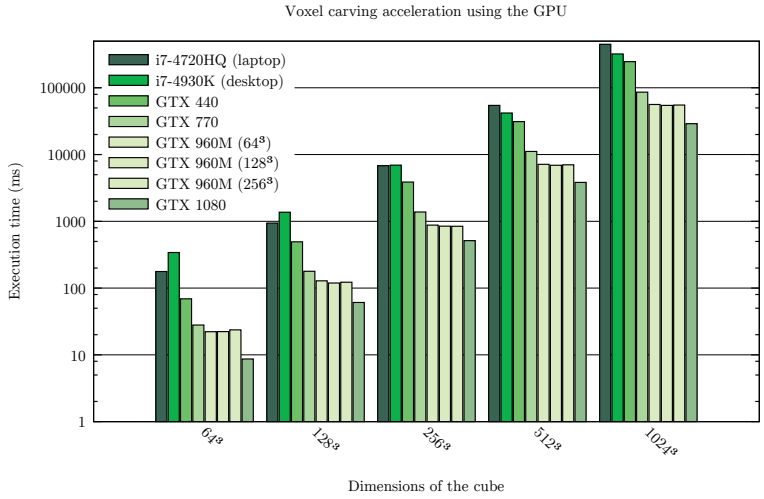


Figure 9.16: The running time for voxel carving from 7 input images, on a variety of processing platforms - two CPUs and 4 GPUs. The three variants for the GTX960M are due to the memory restrictions: it is hard to fit one 1024^3 *cube into memory for low-end cards, and so we perform the voxel carving block-by-block with a given block size. The choice of the block size does not make much difference.*

As we have already mentioned, the cubic nature of the dense volumetric representation requires a large amount of memory and computations. However, the strong degree of independence between neighboring voxels makes this problem a prime candidate for large-scale parallelization on a GPU.

Using the Quasar language [Goossens 2018], we have implemented the voxel carving algorithm on a CPU and on various GPUs. The language framework automatically sets several parameters that are traditionally cumbersome to tweak, selecting the optimal values for the GPU the program is being run on. This allows us to test the performance on a large range of cards, and evaluate the throughput [Donné 2016d]. The results are shown in figure 9.16. The GTX1080, a high-end consumer-level GPU, achieves a speed-up factor of 20-40 over the CPU – a marked increase in speed.

As the very large voxel cubes can pose a memory issue for low-end cards, we have also implemented block-wise carving of the voxel cube, evaluating three block sizes. Here, we sequentially launch one GPU kernel at a time to process each block. This implementational choice does not noticeably impact the performance of the method, as seen in figure 9.16.

Recall that the BAHAMAS project is somewhat time-constrained. We allocate 30 seconds to perform the shape-from-silhouettes 3D reconstruction. The segmentation of the images is in this case non-trivial and required a deep learning network

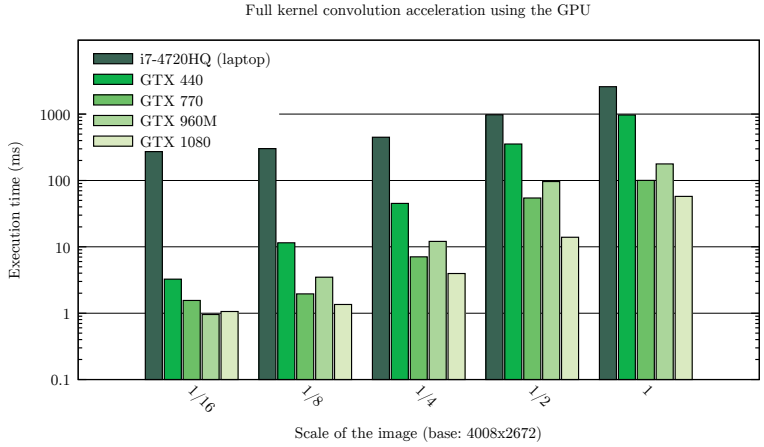


Figure 9.17: The segmentation of the input images also needs to be done as quickly as possible – a task that also greatly benefits from the speed-up provided by GPUs. Here we have evaluated the running time for segmenting a single image. The x-axis expresses the size of the input image, as a fraction of the full size of 4008×2672.

as discussed in section 9.4.1, which is also to happen within that same time constraint. As the segmentation was implemented as a fully convolutional network, it also strongly benefited from the GPU cards for their speed-up, as shown in figure 9.17. This task benefits even more from the speed-up of GPUs, as it is more computationally heavy and less restricted by memory speed as the voxel carving was: there, the evaluation of a single voxel boils down to projecting it on each image (evaluating a set of rational equations) and then performing the relevant memory look-up.

The GTX1080 boasts a speed-up factor of roughly 100 for the segmentation case, and a speed-up factor of 20-40 for the voxel carving – impressive numbers. The result of this all is that, with the GTX1080 at our disposal, we can perform segmentation of all 7 input images at full scale as well as the voxel carving at a resolution of 1024^3 within the allotted 30 seconds. To put this into perspective, if we were restricted to the CPU, we would only be able to process the input images at half their original size, and would still be restricted to carving a 256^2 cube.

9.4.4 Plant modeling

Recall that the end goal of the project is to perform automatic measurements of the maize plants. Specifically, we wish to easily measure the length of a leaf and its surface area; the length of a leaf is measured from its top to the ground, while

the surface area is measured from the tip to the stem. This is hard to do from the semantically meaningless occupancy grid we have estimated above; instead, we now create a skeleton model of the maize plant based on this occupancy grid.

Basic skeletonization techniques exist [Svensson 2002], but they are still troubled by spurious artifacts, and do not take the unique characteristics of maize plants into account. Instead, we model the maize plant as a roughly vertical stem growing from the center of the (known) plant pot, from which several offshoots sprout. First we fit a central stem to the plant model, and afterwards we model the offshoots splitting off from that stem mathematically.

We start by discussing the potential field on which our optimization algorithms will act: the binary occupancy field only yields very quantized and low-range gradients directly near the borders of occupied areas. Afterwards, we outline the optimization procedures for fitting the various parts of the plants to this potential field. All the discussions below involve the locations of the stem, offshoots, tips, etc., in voxel coordinates (d, h, w). As has been mentioned above, the conversion from voxel coordinates to world coordinates is given by

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 + (d + 1/2)s_x \\ y_0 + (h + 1/2)s_y \\ z_0 + (w + 1/2)s_z \\ 1 \end{bmatrix}, \tag{9.4}$$

indicating that we always assume axis-aligned voxel grids.

9.4.4.1 The potential field

As outlined above, we will be optimizing the position, shape and orientation of a variety of geometrical objects based on the occupancy grid we estimated through shape-from-silhouettes. However, that occupancy grid is binary: a voxel is either occupied or it is not. Therefore the use of the occupancy grid as a potential field, from which we calculate gradients, is impractical: the gradients are heavily quantized and their spatial reach is limited. Instead, we create a potential field $\mathcal{D}(d, h, w)$ for optimization by convolving the occupancy grid $\mathcal{O}(d, h, w)$ with a Gaussian blur kernel G_σ:

$$\mathcal{D}(d, h, w) = [G_\sigma * \mathcal{O}](d, h, w) \tag{9.5}$$

The variance of the blur kernel we choose to be half the width of a typical stem, which is about 5 voxels in the case of a $256 \times 256 \times 256$. Figure 9.18 illustrates the impact on a single horizontal slice of the plant stem. As the values of the occupancy grid are binary, and either 0 or 1, the values of the potential field lie in that same interval.

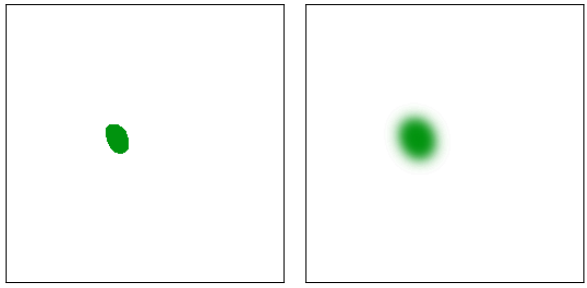


Figure 9.18: Slice of the transformation of the occupancy field (left) to a potential field (right). After this, the gradients on the potential field are much less quantized and affect a much larger area, expediting the optimization procedures.

9.4.4.2 Modeling the stem

The stem is modeled in two steps: first, we find its bottom (where it enters the soil), and afterwards we find its top (where it becomes a leaf). Note that the end of the stem and the beginning of the top leaf (which is often roughly vertical) is even visually hard to detect, as can be seen in the previous visual-spectrum images. However, this precise transition point is not of large importance: biotechnological researchers express the length of a leaf from its tip to the soil, rather than from the tip of the leaf to the stem.

We find the starting point of the stem based on our calibration of the system: all cameras are fully calibrated, and the height of the conveyor belt elevator is stored in the meta-information of the image files. From this, we can extract the 3D location for the middle of the soil in the pot, serving as our initial estimate $\boldsymbol{p}_s^{(0)}$ of the stem start location $\boldsymbol{p}_s$. Because the maize plant was not necessarily planted in the exact center of the pot, this location is optimized locally by performing gradient ascent in the relevant horizontal slice of the occupancy field. The estimate is iteratively optimized by following the local gradient:

$$\boldsymbol{p}_s^{(j+1)} = \boldsymbol{p}_s^{(j)} + \tau \boldsymbol{\nabla} \mathcal{D}\left(\boldsymbol{p}_s^{(j)}\right), \tag{9.6}$$

where the off-grid values of $\mathcal{D}$ are obtained using linear interpolation. The step-size τ is a tuneable parameter, which should be chosen small enough so as not to step past the optimum. We have found $\tau = 0.1$ to be a good choice. Using this update rule, we iterate until convergence to yield $\boldsymbol{s}$. This is illustrated in figure 9.19.

Given the stem start location $\boldsymbol{p}_s$, we now require the end point of the stem – the entire stem is well represented with the line segment between both. From the meta data of the images, we also know the total height of the plant ℓ. The stem is straight as well as nearly – but not completely – vertical. Now, we expand the

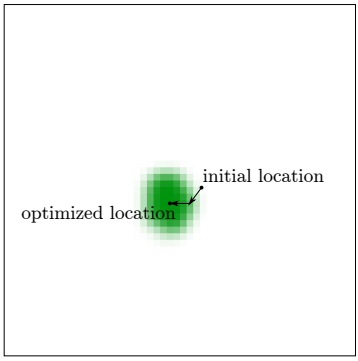


Figure 9.19: Visualization of the stem start location optimization. We know the height of the stem start from the robot meta data. The position in the horizontal plane is estimated using gradient ascent on a blurred version of the relevant occupancy grid slice.

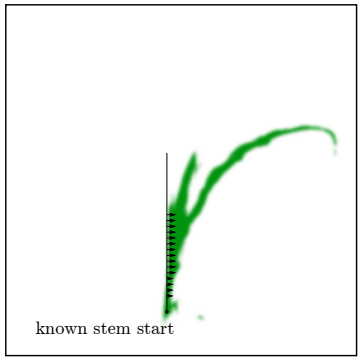


Figure 9.20: Visualization of the stem direction optimization. We initialize the stem direction as being vertical, with a height known from the robot meta data (a height scanner is used at the entrance to the imaging cabin). After that, it is optimized using gradient ascent, illustrated here in 2D.

single-point optimization from equation (9.6) to the optimization of a line segment, with one end fixed. Rather than optimizing the stem direction using the full height measured by the height scanner, we do this based on half that length: towards the top of the plant, the offshoots influence the optimization too much, and some of the offshoots grow higher than the top of the stem. The halfway point of the stem is called $\boldsymbol{p}_m$, with the top of the stem denoted by $\boldsymbol{p}_e$.

Say we have a line segment $\boldsymbol{s}(t)$ with one known end at $\boldsymbol{p}_s$ and with a known height $\ell/2$; we call the other end of the segment $\boldsymbol{p}_m$. Then, the line segment is described by

$$\boldsymbol{s}(t) = \boldsymbol{p}_s + t(\boldsymbol{p}_m - \boldsymbol{p}_s), \quad t \in [0, 1]. \tag{9.7}$$

The only parameter to optimize in this expression is the end of the segment, $\boldsymbol{p}_m$. To do this, we define the problem as an optimization problem that maximizes the total line segment potential over the line segment direction. However, as the height is known to be ℓ, this restricts the movement of $\boldsymbol{p}_m$ to the horizontal plane $\mathcal{P}$ at distance ℓ from $\boldsymbol{p}_s$:

$$\max_{\boldsymbol{p}_m \in \mathcal{P}} \int_{t=0}^{t=1} \mathcal{D}(\boldsymbol{s}(t)). \tag{9.8}$$

Integrals are impractical to work with, so we discretize the step parameter t into

$T+1$ steps for the final objective function:

$$\max_{\boldsymbol{p}_m \in \mathcal{P}} \Phi(\boldsymbol{p}_m) = \frac{1}{T+1} \sum_{t=0}^{t=T} \mathcal{D}\left(\boldsymbol{s}\left(\frac{t}{T}\right)\right). \tag{9.9}$$

As our initial estimate we take the vertical line segment of given length $\ell/2$:

$$\boldsymbol{p}_m^{(0)} = \boldsymbol{p}_s + \frac{\ell}{2}\boldsymbol{e}_z, \tag{9.10}$$

where $\boldsymbol{e}_z$ is a unit vector denoting the vertical direction, which is dependent on the reference system – we have taken it to be given by the direction of the rotation axis of the robot as estimated in section 9.2.1. The gradient update step without taking the plane constraint $\mathcal{P}$ is then given by:

$$\frac{\partial}{\partial \boldsymbol{p}_m}\Phi(\boldsymbol{p}_m) = \frac{1}{T+1} \sum_{t=0}^{t=T} \frac{t}{T} \boldsymbol{\nabla}\mathcal{D}\left(\boldsymbol{r}\left(\frac{t}{T}\right)\right). \tag{9.11}$$

A visual representation of this gradient accumulation is given in figure 9.20. After constraining the movement of $\boldsymbol{p}_m$ to $\mathcal{P}$ by subtracting the relevant contribution in the partial derivative, the update steps are now given by

$$\boldsymbol{p}_m^{(j+1)} = \boldsymbol{p}_m^{(j)} + \tau \left(\frac{\partial}{\partial \boldsymbol{p}_m^{(j)}}\Phi(\boldsymbol{p}_m^{(j)}) - \left(\boldsymbol{e}_z \cdot \frac{\partial}{\partial \boldsymbol{p}_m^{(j)}}\Phi(\boldsymbol{p}_m^{(j)})\right)\boldsymbol{e}_z \right). \tag{9.12}$$

Here, τ again denotes the step-size, and we again choose it to be equal to 0.1.

After the direction is known, the top of the stem $\boldsymbol{p}_e$ is estimated by simply extending a line along this direction until the point at its tip would leave the occupied space in the voxel carving result.

9.4.4.3 Modeling the offshoots

After estimating the stem position and length, we now wish to model the offshoots. We do this by detecting the tips of the leaves. We spawn a dense grid of tip candidates at every 8th voxel, which are locally optimized with exactly the same procedure as in equation (9.6). After this local optimization, we calculate the direction $\delta\boldsymbol{p}$ of the tip candidate $\boldsymbol{p}$ as the direction from the candidate to the barycenter $\bar{\boldsymbol{p}}$ of the local density volume, a small 17^3 volume around each candidate:

$$\begin{aligned} \bar{\boldsymbol{p}} &= \boldsymbol{p} + \frac{\sum_{x_o,y_o,z_o=-8}^{8} [x_o, y_o, z_o]^{\mathrm{T}}\, \mathcal{D}(\boldsymbol{p} + [x_o, y_o, z_o]^{\mathrm{T}})}{\sum_{x_o,y_o,z_o=-8}^{8} \mathcal{D}(\boldsymbol{p} + [x_o, y_o, z_o]^{\mathrm{T}})} \\ \delta\boldsymbol{p} &= \frac{\bar{\boldsymbol{p}} - \boldsymbol{p}}{\|\bar{\boldsymbol{p}} - \boldsymbol{p}\|_2} \end{aligned} \tag{9.13}$$

In case of ambiguities (the $\bar{\boldsymbol{p}}$ is less than a voxel away from $\boldsymbol{p}$), we simply take the direction to be from the candidate to the middle of the stem. This is a valid approximation because the center-lines of the leaves lie roughly within the plane defined by their tip and the stem, see the top-view in figure 9.3. We now remove any tip candidates that have other candidates behind them, in the sense dictated by their orientation, i.e.

$$\exists \boldsymbol{q} : (\boldsymbol{p} - \boldsymbol{q}) \cdot \delta\boldsymbol{q} \leq -0.5 \implies \boldsymbol{p} \text{ is an invalid candidate.} \tag{9.14}$$

For the remaining candidates, we now trace an optimal path through the occupancy density field starting from the tip. Iteratively, and until we reach the stem, we place a new point at a predefined distance from the previous one along its direction, and then optimize this point locally. At each iteration, we have the current point $\boldsymbol{p}_i$, and its direction $\delta\boldsymbol{p}_i$. We wish to find the next point $\boldsymbol{p}_{i+1}$ along the leaf centerline. We spawn the initial estimate $\boldsymbol{p}_{i+1}^{(0)}$ at a given distance ν from the last one:

$$\boldsymbol{p}_{i+1}^{(0)} = \boldsymbol{p}_i + \nu\delta\boldsymbol{p}_i. \tag{9.15}$$

We have taken $\nu = 2$ voxels in our implementation. Now, we optimize $\boldsymbol{p}_{i+1}$ iteratively, similarly to equation (9.6). However, we restrict its movement in the plane with normal $\delta\boldsymbol{p}_i$, i.e., we restrict its movement to a given cross section of the leaf. Therefore, the update rule is now given by

$$\boldsymbol{p}_{i+1}^{(j+1)} = \boldsymbol{p}_{i+1}^{(j)} + \tau \left(\boldsymbol{\nabla}\mathcal{D}\left(\boldsymbol{p}_{i+1}^{(j)}\right) - \left(\boldsymbol{\nabla}\mathcal{D}\left(\boldsymbol{p}_{i+1}^{(j)}\right) \cdot \delta\boldsymbol{p}_i \right) \delta\boldsymbol{p}_i \right), \tag{9.16}$$

where we have removed that part of the gradient along the orientation of $\boldsymbol{d}$: the point $\boldsymbol{s}$ is only free to move orthogonal to this direction. This is repeated until convergence.

For every point $\boldsymbol{p}_i$ except the first one, the direction $\delta\boldsymbol{p}_i$ cannot be calculated from the barycenter of the surrounding points, as that is likely to be near $\boldsymbol{p}_i$ because we are already roughly on the centerline of the leaf. Therefore, we estimate the directions by taking the major axis of the PCA factorization of the set of offsets to all occupied voxels in the 17^3 neighborhood.

9.4.4.4 Duplicate trajectory filtering

However, there are cases as in figure 9.21 where, because of some noise in the occupancy field, multiple tip candidates are being traced for a single leaf and obviously we want to filter those out. We do this by calculating which fraction of the sample points on the paths lies within a given threshold of the other path. If this is more than half, we assume the paths to be from the same leaf, and we only keep the longest of the two.

Assume we have two trajectories given by the point sequences $\{\boldsymbol{p}_i\}$ and $\{\boldsymbol{q}_j\}$, where $\boldsymbol{p}_0$ and $\boldsymbol{q}_0$ are the points of the trajectories on the stem (where the leaf

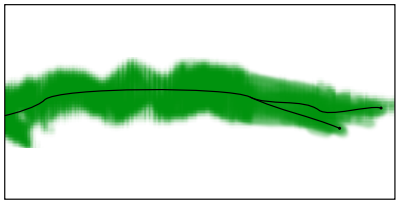


Figure 9.21: Occasionally, artifacts in the occupancy grid cause multiple tip candidates to survive for one leaf.

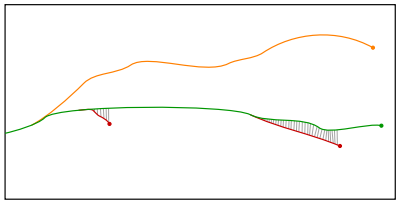


Figure 9.22: The closeness of two paths is dictated by the average distance between the curves.

tracing ends), and $\boldsymbol{p}_I$ respectively $\boldsymbol{q}_J$ are the tips of the leaves. We adopt the following dynamic programming procedure to quantify the distance between both curves:

$\delta \leftarrow \|\boldsymbol{p}_0 - \boldsymbol{q}_0\|_2$
$i \leftarrow 0,\ j \leftarrow 0$
while $i < I$ **and** $j < J$ **do**
    if $\|\boldsymbol{p}_{i+1} - \boldsymbol{q}_j\|_2 < \|\boldsymbol{p}_i - \boldsymbol{q}_{j+1}\|_2$ **then**
        $i \leftarrow i + 1$
        $\delta \leftarrow \delta + \|\boldsymbol{p}_{i+1} - \boldsymbol{q}_j\|_2$
    else
        $j \leftarrow j + 1$
        $\delta \leftarrow \delta + \|\boldsymbol{p}_i - \boldsymbol{q}_{j+1}\|_2$
    end if
end while
$\delta \leftarrow \delta/(i + j)$

This procedure maintains two points, one along each curve. It moves those points alternately along their respective curve, based on which movement would entail the smallest distance to the (fixed) point on the other curve – in this way we are robust against differing sampling intervals of the curves. This is done until the end of either curve is reached – in this way, we can easily handle one curve being much shorter than the other. If this inter-curve distance δ is too low (smaller than the roughly-leaf-width threshold of 4 voxels), we discard the shortest of both trajectories.

9.4.4.5 Leaf modeling as Beziers

Finally, we discuss the fitting of a Bezier curve to the points $\{\boldsymbol{p}_i\}$. A Bezier curve $\boldsymbol{b}(t)$ is a smooth curve defined by $N+1$ points $\boldsymbol{g}_n$: a start and end point, and $N-1$ control points. Its equation is a polynomial combination of the control points:

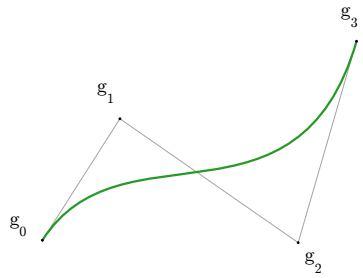


Figure 9.23: 2D example of a Bezier curve. Starting in $\boldsymbol{g}_0$ and ending in $\boldsymbol{g}_N$, the intermediate points get pulled to each of the guide points in sequence, resulting in a smooth and natural-looking trajectory with only a few intuitive parameters.

$$\begin{aligned} \boldsymbol{b}(t) &= \sum_{n=0}^{n=N} \binom{N}{n} (1-t)^{N-n} t^n \boldsymbol{g}_n, \text{ where} \\ \binom{N}{n} &= \frac{N!}{n!(N-n)!}. \end{aligned} \tag{9.17}$$

Note that this expression entails $\boldsymbol{b}(0) = \boldsymbol{g}_0$ and $\boldsymbol{b}(1) = \boldsymbol{g}_N$. In between those points, the curve gets pulled towards the various control points, as illustrated in figure 9.23.

As in the stem direction optimization, we define the optimization problem to be the maximization of the potential of this curve over the control points:

$$\max_{\{\boldsymbol{g}_n\}} \int_{t=0}^{t=1} \mathcal{D}(\boldsymbol{b}(t)), \tag{9.18}$$

where we reformulate the integral again as a discrete sum of $T+1$ elements:

$$\max_{\{\boldsymbol{g}_n\}} \Phi(\{\boldsymbol{g}_n\}) = \frac{1}{T+1} \sum_{t=0}^{t=T} \mathcal{D}\left(\boldsymbol{b}\left(\frac{t}{T}\right)\right). \tag{9.19}$$

Initial Bezier curve estimate

As in the previous optimization problems, we require an initial estimate of the Bezier curve. Sadly, this is less straightforward now, as we do not know the various t values corresponding to the measured sample points $\boldsymbol{p}_i$. However, we fix the beginning and end point as respectively the first and last point in the trajectory. First we note that, if we were to know the t_i value for each point, the problem would boil down to solving (in a minimum-square sense) the system

$$\boldsymbol{b}(t_i) = \boldsymbol{p}_i, \quad \forall i. \tag{9.20}$$

However, the values $\{t_i\}$ are also unknown and need to be estimated. We will tackle this in an iterative way, too. The initial estimate is given by the fractional position along the trajectory:

$$t_i^{(0)} = \frac{\sum_{j=1}^{j=i} \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|_2}{\sum_{j=1}^{j=I} \|\boldsymbol{p}_j - \boldsymbol{p}_{j-1}\|_2}. \tag{9.21}$$

This is a fair initial estimate, but not completely accurate because this is not the real meaning of t_i – specifically in segments of the curve with high curvature, this estimate will be noticeably off. Now, we optimize the Bezier curve fitting iteratively by switching between optimizing the control points and the t_i values. In subsequent steps, we estimate t_i to be the t value of the closest point on the Bezier curve:

$$t_i(\{\boldsymbol{g}_n\}) = \underset{t}{\operatorname{argmin}} \|\boldsymbol{p}_i - \boldsymbol{b}(t)\|_2. \tag{9.22}$$

This results in the following procedure for fitting a Bezier curve to the given set of points.

1: $t_i^{(0)} \leftarrow$ initialization from equation (9.21)
2: $\boldsymbol{g}_n^{(0)} \leftarrow$ MSE solution to equation (9.20)
3: **repeat**
4: $\quad t_i^{(j+1)} \leftarrow$ solution to equation (9.22)
5: $\quad \boldsymbol{g}_n^{(j+1)} \leftarrow$ MSE solution to equation (9.20)
6: **until** convergence

Refining the estimate

After obtaining a reasonable initialization of the Bezier curve based on the trajectory points, we still wish to optimize the control points of the curve directly guided by the potential field. Again, we keep the start (leaf tip) and end (connection to the stem) of the Bezier curve fixed. This means we only optimize the location of the intermediate points $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{N-1}$. We can calculate the derivative of the cost function $\Phi(\{\boldsymbol{g}_n\})$ with respect to the various control points using the chain rule:

$$\begin{aligned} \frac{\partial}{\partial \boldsymbol{g}_n}\Phi(\{\boldsymbol{g}_n\}) &= \frac{1}{T+1}\sum_{t=0}^{t=T} \boldsymbol{\nabla}\mathcal{D}\left(\boldsymbol{b}\left(\frac{t}{T}\right)\right)\frac{\partial}{\partial \boldsymbol{g}_n}\boldsymbol{b}\left(\frac{t}{T}\right), \text{ with} \\ \frac{\partial}{\partial \boldsymbol{g}_n}\boldsymbol{b}\left(\frac{t}{T}\right) &= \binom{N}{n}\left(1-\frac{t}{T}\right)^{N-n}\left(\frac{t}{T}\right)^{n}. \end{aligned} \tag{9.23}$$

The update rules are again given by simple gradient ascent (we choose $\tau = 0.1$ again):

$$\boldsymbol{g}_n^{(j+1)} = \boldsymbol{g}_n^{(j)} + \tau\frac{\partial}{\partial \boldsymbol{g}_n}\Phi(\{\boldsymbol{g}_n\}). \tag{9.24}$$

Figure 9.24: The result of the 3D plant modeling: from the estimated occupancy grid, we have extracted the stem of the plant as a line segment, and each of the leaves as a Bezier curve.

9.5 Automatic measurements

After all these steps, we arrive at a plant model comprising the estimated occupancy grid, the stem line segment and a set of Bezier curves, one for each leaf. This is visualized in figure 9.24. At long last, we can progress with the automatic measurements of relevant entities.

Recall that we wish to extract both the leaf lengths and the leaf areas. A leaf's length follows immediately from the 3D model of the maize plant we already have: we can simply measure the length of the Bezier curve, again by discretization of the involved integral, plus the distance from its connection with the stem to the ground:

$$\sum_{t=1}^{t=T} \left\| \boldsymbol{b}\left(\frac{t}{T}\right) - \boldsymbol{b}\left(\frac{t-1}{T}\right) \right\|_2 + \|\boldsymbol{b}(0) - \boldsymbol{p}_s\|_2 \tag{9.25}$$

The leaf area is calculated in terms of voxel faces. We collect, for each leaf, the occupied voxels that are closer to this leaf than to another. Each voxel that does not have an occupied voxel directly above it is considered a surface voxel, and we count those. This yields us an estimate of the leaf area, and in a natural way filters out those voxels belonging to the stem of the plant.

9.6 Conclusion

In this chapter, we have shown how to incorporate several previously discussed techniques in the full pipeline of a biotechnological project. From camera calibration to efficient 3D reconstruction, several of our research topics throughout this thesis were applied to a practical problem, building an automated measurement system for maize plants to ease future biotechnological research. In this specific light, several works have been published focused on the biotechnological application:

- *Machine Learning for Maize Plant Segmentation*, Simon Donné, Hiep Luong, Bart Goossens, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé and Wilfried Philips. The Belgian-Dutch Conference on Machine Learning, BENELEARN 2016 [Donné 2016e].
- *GPU-based maize plant analysis: accelerating CNN segmentation and voxel carving*, Simon Donné, Bart Goossens, Stijn Dhondt, Nathalie Wuyts, Hiep Luong, Dirk Inzé and Wilfried Philips. NVIDIA European GPU Technology Conference, NVIDIA EGTC 2016 [Donné 2016c].
- *3D Reconstruction of maize plants in the PhenoVision system*, Simon Donné, Hiep Luong, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé and Wilfried Philips. Knowledge for Growth, KfG 2016 [Donné 2016d].

The PhenoVision robot makes large-scale, detailed, plant analysis a possibility. From the perspective of the biotechnological research, this gives access to a treasure trove of information that was previously not available in such quantities, as well as a direct link to the extra modalities (such as the VNIR and SWIR cameras). Using all of this detailed information efficiently will be a challenge in its own right.

10
Closing Remarks

" Now these points of data make a beautiful line,
and we're out of beta, we're releasing on time. "

— Jonathan Coulton, *Portal*

Imaging systems are constantly improving: ever more precisely constructed cameras capture increasingly complex scenes at constantly increasing resolutions. They are no longer restricted to the traditional red-green-blue channels but now acquire many different modalities, each with their own advantages and peculiarities. Computer vision researchers, and the algorithms they develop, need to keep up with these advancements. In this doctoral work, we have studied the problem of 3D reconstruction: how can the 3D structure of the world around us be reconstructed based on observations by multiple cameras?

While many of the basic building blocks, such as perspective camera calibration or checkerboard detection, are currently well-understood and considered solved to some extent, 3D reconstruction in general is far from a static field. Novel approaches and techniques are constantly being developed, and the performance of these techniques continuously improves. As in other research fields, deep learning techniques are starting to gain ground in 3D reconstruction. Manually designing algorithms for computer vision, in a bid to emulate the human brain, is prone to carry our prejudices and biases into the algorithms. We have all occasionally been fooled by optic illusions of some kind: the human brain is not infallible and handcrafting algorithms for scene interpretation could introduce the some loopholes in our algorithms. Deep learning, which automatically infers its reasoning from a set

of training data and ground truth, is assumed to be less susceptible to these loopholes. At the same time, there are also many aspects of computer vision, and image processing in general, where we know the formal rules dictating the system's behavior very well. Think of the point triangulation in chapter 6 or the camera calibration from chapter 3: for such problems, deep learning has not even come close to replicating the performance of hand-crafted techniques. Deep learning is not the be-all and end-all solution to our every computer vision problem, but it does offer many advantages in a plethora of cases. In this work, the role of deep learning was restricted to fields in which it is already well established, such as pattern recognition (for the checkerboard detection) or foreground-background segmentation (for the maize plant images), but recent work on 3D reconstruction with deep learning is implying that soon it will also find applications in the 2D-to-3D reconstruction step itself [Ji 2017, Kar 2017].

The structure of this work was organized to reflect the entire workflow of a computer vision project for object reconstruction. Concluding with chapter 9, it shows how intricately every part of the chain is linked, and how we combine the many aspects to solve practical problems. In the body of this manuscript, we have discussed many different improvements to the preexisting literatures, framing each in the field of computer vision:

- Hyperspectral cameras, which image many more wavelengths compared to traditional cameras, do not follow the traditional perspective camera model. The existing work on calibrating these had important flaws with respect to certain common cases, which we have addressed in [Donné 2017b].

- We have proposed a deep learning approach to detecting checkerboards in images [Donné 2016b]. We show that it matches or improves the performance of preexisting techniques, while also yielding good results on different types of patterns after retraining. Going forward, this flexibility allows us to apply this technique on different modalities or calibration patterns.

- We have proposed a novel geometrically-inspired optimization technique for ℓ_∞ point triangulation [Donné 2015c]. Aside from being faster than existing literature, it also enables an intuitive interpretation of its workings and can be easily implemented on a GPU.

- Our keyframe selection algorithm for history representation enables the online application of non-rigid structure-from-motion in long video sequences, so that it can be applied to video conferencing [Donné 2014].

- Focusing on the computational cost of stereo matching algorithms, we have investigated the impact and benefit of oversegmentation for speeding up stereo matching algorithms [Donné 2015a].

– We have also proposed the use of an extended version of the binocular stereo matching case for multi-view stereo matching [Donné 2015b, Donné 2017a]. At the cost of a less general set-up, it strongly alleviates the issue of occlusions in binocular stereo matching.

– We have investigated the efficiency of various algorithms on the GPU: voxel carving [Donné 2016c], image segmentation [Donné 2016e], and physics simulations [Donné 2016a] (the latter was not explicitly discussed in this thesis).

– Finally, we have combined many of the above approaches into a complete pipeline for automatic maize plant segmentation [Donné 2016d].

References

[Achanta 2010] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua & Sabine Süsstrunk. *SLIC Superpixels*. Technical report, EPFL, 2010.

[Achanta 2012] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurélien Lucchi, Pascal Fua & Sabine Süsstrunk. *SLIC Superpixels Compared to State-of-the-art Superpixel Methods*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 34, no. 11, pages 2274–2282, 2012.

[Adelson 1992] Edward H. Adelson & John Y. A. Wang. *Single lens stereo with a plenoptic camera*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 14, no. 2, pages 99–106, feb 1992.

[Agarwal 2008] Sameer Agarwal, Noah Snavely & Steven M. Seitz. *Fast algorithms for L-infinity problems in multiview geometry*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), jun 2008.

[Agrawal 2003] Motilal Agrawal & Larry S Davis. *Camera calibration using spheres: A semi-definite programming approach*. In IEEE International Conference on Computer Vision (ICCV), pages 782–789, 2003.

[Akhter 2009] Ijaz Akhter, Yaser Sheikh, Sohaib Khan & Takeo Kanade. *Nonrigid Structure from Motion in Trajectory Space*. In Advances in Neural Information Processing Systems (NIPS), pages 41–48, 2009.

[Akhter 2011] Ijaz Akhter, Yaser Sheikh, Sohaib Khan & Takeo Kanade. *Trajectory Space: A Dual Representation for Nonrigid Structure from Motion*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 33, no. 7, pages 1442–1456, jul 2011.

[Arca 2005] Stefano Arca, Elena Casiraghi & Gabriele Lombardi. *Corner localization in chessboards for camera calibration*. In International Conference on Multimedia, Image Processing and Computer Vision (ICMIPC), 2005.

[Bailer 2012] Christian Bailer, Manuel Finckh & Hendrik P.A. Lensch. *Scale robust multi view stereo*. In Lecture Notes in Computer Science, pages 398–411, 2012.

[Bailer 2015] Christian Bailer, Bertram Taetz & Didier Stricker. *Flow Fields: Dense Correspondence Fields for Highly Accurate Large Displacement Optical Flow Estimation*. In IEEE International Conference on Computer Vision (ICCV), pages 4015–4023, 2015.

[Barreto 2006] João Barreto & Kostas Daniilidis. *Epipolar geometry of central projection systems using veronese maps*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1258–1265, 2006.

[Baumgart 1974] Bruce Günther Baumgart. *Geometric modeling for computer vision*. PhD thesis, Stanford University, 1974.

[Bay 2008] Herbert Bay, Andreas Ess, Tuytelaars Tinne & Luc Van Gool. *Speeded-Up Robust Features (SURF)*. Computer Vision and Image Understanding, vol. 110, no. 3, pages 346–359, 2008.

[Bayer 1976] Bryce E. Bayer. *Color imaging array*, 1976.

[Bell 1989] Sarah Bell, Fred Holroyd & David Mason. *A digital geometry for hexagonal pixels*. Image and Vision Computing, vol. 7, no. 3, pages 194–204, 1989.

[Bennett 2014] Stuart Bennett & Joan Lasenby. *ChESS – Quick and robust detection of chess-board features*. Computer Vision and Image Understanding, vol. 118, pages 197–210, 2014.

[Bhotika 2002] Rahul Bhotika, David J Fleet & Kiriakos N Kutulakos. *A probabilistic theory of occupancy and emptiness*. In IEEE European Conference on Computer Vision (ECCV), pages 112–130, 2002.

[Bishop 2007] Christopher M Bishop. Pattern Recognition and Machine Learning. Springer, 2007.

[Blanz 1999] Volker Blanz & Thomas Vetter. *A morphable model for the synthesis of 3D faces*. In Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 187–194, 1999.

[Bok 2016] Yunsu Bok, Hyowon Ha & In So Kweon. *Automated Checkerboard Detection and Indexing using Circular Boundaries*. Pattern Recognition Letters, vol. 71, pages 66–72, 2016.

[Bottou 2010] Léon Bottou. *Large-scale machine learning with stochastic gradient descent*. In International Conference on Computational Statistics (COMPSTAT), pages 177–186, 2010.

[Bouguet 2004] Jean-Yves Bouguet. *Camera calibration toolbox for matlab*, 2004.

[Bouwmans 2011] Thierry Bouwmans. *Recent advanced statistical background modeling for foreground detection-a systematic survey*. Recent Patents on Computer Science, vol. 4, no. 3, pages 147–176, 2011.

[Bredies 2010] Kristian Bredies, Karl Kunisch & Thomas Pock. *Total generalized variation*. SIAM Journal on Imaging Sciences, vol. 3, pages 492–526, 2010.

[Bregler 2000] Christoph Bregler, Aaron Hertzmann & Henning Biermann. *Recovering non-rigid 3D shape from image streams*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 690–696, 2000.

[Brown 1966] Duane C. Brown. *Decentering distortion of lenses*. Photogrammetric Engineering and Remote Sensing, 1966.

[Byröd 2007] Martin Byröd, Klas Josephson & Kalle Åström. *Fast Optimal Three View Triangulation*. In Yasushi Yagi, SingBing Kang, InSo Kweon & Hongbin Zha, editeurs, IEEE Asian Conference on Computer Vision (ACCV), volume 4844 of *Lecture Notes in Computer Science*, pages 549–559, 2007.

[Calonder 2010] Michael Calonder, Vincent Lepetit, Christoph Strecha & Pascal Fua. *BRIEF: Binary Robust Independent Elementary Features*. In IEEE European Conference on Computer Vision (ECCV), pages 778–792, 2010.

[Chambolle 2011] Antonin Chambolle & Thomas Pock. *A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging*. Journal of Mathematical Imaging and Vision, vol. 40, no. 1, pages 120–145, 2011.

[Chang 2003] Chein-I Chang. Hyperspectral imaging: techniques for spectral detection and classification. Springer Science & Business Media, 2003.

[Chang 2013] Haw-Shiuan Chang & Yu-Chiang Frank Wang. *Superpixel-based large displacement optical flow*. In IEEE International Conference on Image Processing (ICIP), sep 2013.

[Chen 2015] Zhuoyuan Chen, Xun Sun, Liang Wang, Yinan Yu & Chang Huang. *A Deep Visual Correspondence Embedding Model for Stereo Matching Costs*. In IEEE International Conference on Computer Vision (ICCV), pages 972–980, dec 2015.

[Collins 1996] Robert T Collins. *A space-sweep approach to true multi-image matching*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 358–363, 1996.

[Conrady 1919] Alexander Eugen Conrady. *Decentered Lens-systems*. Monthly notices of the royal astronomical society, vol. 79, pages 384–390, 1919.

[Curless 1996] Brian Curless & Marc Levoy. *A volumetric method for building complex models from range images*. In ACM Conference on Computer Graphics and Interactive Techniques, pages 303–312, 1996.

[Dai 2012] Zhijun Dai, Yihong Wu, Fengjun Zhang & Hongan Wang. *A Novel Fast Method for L-infinity Problems in Multiview Geometry*. In IEEE European Conference on Computer Vision (ECCV), volume 7576 of *Lecture Notes in Computer Science*, pages 116–129, 2012.

[Dai 2014] Yuchao Dai, Hongdong Li & Mingyi He. *A simple prior-free method for non-rigid structure-from-motion factorization*. International Journal of Computer Vision (IJCV), vol. 107, no. 2, pages 101–122, 2014.

[Daucher 1994] Nadine Daucher, Michel Dhome & Jean-Thierry Lapresté. *Camera calibration from spheres images*. In IEEE European Conference on Computer Vision (ECCV), pages 447–454, 1994.

[De la Escalera 2010] Arturo De la Escalera & Jose Maria Armingol. *Automatic chessboard detection for intrinsic and extrinsic camera parameter calibration*. MDPI Sensors, vol. 10, no. 3, pages 2027–2044, 2010.

[De Villiers 2008] Jason P. De Villiers, F. Wilhelm Leuschner & Ronelle Geldenhuys. *Centi-pixel accurate real-time inverse distortion correction*. In International Symposium on Optomechatronic Technologies, 2008.

[Dempster 1977] Arthur P. Dempster, Nan M. Laird & Donald B Rubin. *Maximum likelihood from incomplete data via the EM algorithm*. Journal of the Royal Statistical Society, pages 1–38, 1977.

[Dong 2014] Chao Dong, Chen Change Loy, Kaiming He & Xiaoou Tang. *Learning a deep convolutional network for image super-resolution*. In IEEE European Conference on Computer Vision (ECCV), pages 184–199, 2014.

[Dong 2016] Chao Dong, Chen Change Loy & Xiaoou Tang. *Accelerating the super-resolution convolutional neural network*. In IEEE European Conference on Computer Vision (ECCV), pages 391–407, 2016.

[Donné 2014] Simon Donné, Ljubomir Jovanov, Bart Goossens, Wilfried Philips & Aleksandra Pižurica. *Online non-rigid structure-from-motion based on a keyframe representation of history*. In International Conference on Computer Vision Theory and Applications (VISAPP), volume 2, 2014.

[Donné 2015a] Simon Donné, Jan Aelterman, Bart Goossens & Wilfried Philips. *Fast and robust variational optical flow for high-resolution images using SLIC superpixels*. In Advanced Concepts for Intelligent Vision Systems (ACIVS), volume 9386, 2015.

[Donné 2015b] Simon Donné, Bart Goossens, Jan Aelterman & Wilfried Philips. *Variational multi-image stereo matching*.

In IEEE International Conference on Image Processing (ICIP), 2015.

[Donné 2015c] Simon Donné, Bart Goossens & Wilfried Philips. *Point triangulation through polyhedron collapse using the $\ell\infty$ norm*. In IEEE International Conference on Computer Vision (ICCV), 2015.

[Donné 2016a] Simon Donné. *Efficiently simulating n-body systems*. In HiPEAC Fall Computing Week student challenge, 2016.

[Donné 2016b] Simon Donné, Jonas De Vylder, Bart Goossens & Wilfried Philips. *MATE: Machine learning for adaptive calibration template detection*. MDPI Sensors, vol. 16, no. 11, 2016.

[Donné 2016c] Simon Donné, Bart Goossens, Stijn Dhondt, Nathalie Wuyts, Hiep Luong, Dirk Inzé & Wilfried Philips. *GPU-based maize plant analysis: accelerating CNN segmentation and voxel carving*. In NVIDIA European GPU Technology Conference (NVIDIA GTC), 2016.

[Donné 2016d] Simon Donné, Hiep Luong, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé & Wilfried Philips. *3D Reconstruction of Maize Plants in the Phenovision System*. In Knowledge For Growth (KfG), 2016.

[Donné 2016e] Simon Donné, Hiep Luong, Bart Goossens, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé & Wilfried Philips. *Machine Learning for Maize Plant Segmentation*. In Belgian-Dutch Conference on Machine Learning (BENELEARN), 2016.

[Donné 2017a] Simon Donné, Bart Goossens & Wilfried Philips. *Line-constrained camera location estimation in multi-image stereomatching*. MDPI Sensors, vol. 17, no. 9, 2017.

[Donné 2017b] Simon Donné, Hiep Luong, Stijn Dhondt, Nathalie Wuyts, Dirk Inzé, Bart Goossens & Wilfried Philips. *Robust Plane-based Calibration for linear cameras*. In IEEE International Conference on Image Processing (ICIP), 2017.

[Donné 2017c] Simon Donné, Laurens Meeus, Hiep Quang Luong, Bart Goossens & Wilfried Philips. *Exploiting Reflectional and Rotational Invariance in Single Image Superresolution*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[Dosovitskiy 2015] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers & Thomas Brox. *FlowNet: Learning Optical Flow with Convolutional Networks*. In IEEE International Conference on Computer Vision (ICCV), 2015.

[Douterloigne 2013] Koen Douterloigne, Werner Goeman, Sidharta Gautama & Wilfried Philips. *Automatic detection of a one dimensional ranging pole for robust external camera calibration in mobile mapping*. ISPRS Journal of Photogrammetry and Remote Sensing, vol. 86, pages 111–123, 2013.

[Draréni 2011] Jamil Draréni, Sébastien Roy & Peter Sturm. *Plane-based calibration for linear cameras*. International Journal of Computer Vision (IJCV), vol. 91, no. 2, pages 146–156, 2011.

[Drory 2014] Amnon Drory, Carsten Haubold, Shai Avidan & Fred A. Hamprecht. *Semi-Global Matching: A Principled Derivation in Terms of Message Passing*. In German Conference on Pattern Recognition (GCPR), pages 43–53, 2014.

[Drulea 2013] Marius Drulea & Sergiu Nedevschi. *Motion estimation using the correlation transform*. IEEE Transactions on Image Processing, vol. 22, no. 8, pages 3260–3270, 2013.

[Endres 2012] Felix Endres, Jurgen Hess, Nikolas Engelhard, Jurgen Sturm, Daniel Cremers & Wolfram Burgard. *An evaluation of the RGB-D SLAM system*. In IEEE International Conference on Robotics and Automation (ICRA), pages 1691–1696, may 2012.

[Engel 2014] Jakob Engel, Thomas Schöps & Daniel Cremers. *LSD-SLAM: Large-Scale Direct Monocular SLAM*. In IEEE European Conference on Computer Vision (ECCV), pages 834–849, 2014.

[Enqvist 2010] Olof Enqvist, Carl Olsson & Fredrik Kahl. *Stable structure from motion using rotational consistency*. Technical report, Centre for Mathematical Sciences, Lund University, Sweden, 2010.

[Eriksson 2014] Anders Eriksson & Mats Isaksson. *Pseudoconvex Proximal Splitting for L-infinity Problems in Multiview Geometry*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4066–4073, jun 2014.

[Faugeras 1998] Olivier Faugeras & Renaud Keriven. *Variational principles, surface evolution, PDE's, level set methods and the stereo problem*. IEEE Transactions on Image Processing, vol. 7, no. 3, pages 336–344, 1998.

[Fayad 2009] João Fayad, Alessio Del Bue, Lourdes Agapito & Pedro Aguiar. *Non-rigid Structure from Motion using Quadratic Deformation Models.* In British Machine Vision Conference (BMVC), volume 1, page 2, 2009.

[Felzenszwalb 2004] Pedro F. Felzenszwalb & Daniel P. Huttenlocher. *Efficient Graph-Based Image Segmentation*. International Journal of Computer Vision (IJCV), vol. 59, no. 2, pages 167–181, 2004.

[Feng 2013] Jiashi Feng, Huan Xu & Shuicheng Yan. *Online Robust PCA via Stochastic Optimization*. In Advances in Neural Information Processing Systems (NIPS), pages 404–412, 2013.

[Fischer 2006] Christian Fischer & Ioanna Kakoulli. *Multispectral and hyperspectral imaging technologies in conservation: current research and potential applications*. Studies in Conservation, vol. 51, pages 3–16, 2006.

[Forsyth 2011] David Forsyth & Jean Ponce. Computer vision: a modern approach. Upper Saddle River, NJ; London: Prentice Hall, 2011.

[Furukawa 2010a] Yasutaka Furukawa, Brian Curless, Steven M. Seitz & Richard Szeliski. *Towards internet-scale multi-view stereo*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1434–1441. IEEE, jun 2010.

[Furukawa 2010b] Yasutaka Furukawa & Jean Ponce. *Accurate, dense, and robust multiview stereopsis*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 32, no. 8, pages 1362–1376, aug 2010.

[Galliani 2015] Silvano Galliani, Katrin Lasinger & Konrad Schindler. *Massively parallel multiview stereopsis by surface normal diffusion*. In IEEE International Conference on Computer Vision (ICCV), volume 2015 Inter, pages 873–881. IEEE, dec 2015.

[Gehrig 2009] Stefan K. Gehrig, Felix Eberli & Thomas Meyer. *A Real-Time Low-Power Stereo Vision Engine Using Semi-Global Matching*. In International Conference on Computer Vision Systems, pages 134–143, oct 2009.

[Georgiev 2009] Todor Georgiev & Andrew Lumsdaine. *Superresolution with plenoptic camera 2.0*. Technical report, Adobe Systems Incorporated, 2009.

[Gkamas 2011] Theodosios Gkamas & Christophoros Nikou. *Guiding optical flow estimation using superpixels*. In International Conference on Digital Signal Processing (DSP), pages 1–6, 2011.

[Gokturk 2004] S. Burak Gokturk, Hakan Yalcin & Cyrus Bamji. *A time-of-flight depth sensor-system description, issues and solutions*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), page 35. IEEE, 2004.

[Goossens 2018] Bart Goossens. *Dataflow Management, Dynamic Load Balancing and Concurrent Processing for Real-time Embedded Vision Applications using Quasar*. International Journal of Circuit Theory and Applications, Special Issue on Computational Image Sensors and Smart Camera Hardware, 2018.

[Gotardo 2011a] Paulo F. U. Gotardo & Aleix M. Martinez. *Computing Smooth Time Trajectories for Camera and Deformable Shape in Structure from Motion with Occlusion*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 33, no. 10, pages 2051–2065, oct 2011.

[Gotardo 2011b] Paulo F. U. Gotardo & Aleix M. Martinez. *Kernel non-rigid structure from motion*. In IEEE International Conference on Computer Vision (ICCV), pages 802–809, nov 2011.

[Gower 1975] John C. Gower. *Generalized procrustes analysis*. Psychometrika, vol. 40, no. 1, pages 33–51, 1975.

[Guan 2015] Junzhi Guan, Francis Deboeverie, Maarten Slembrouck, Dirk Van Haerenborgh, Dimitri Van Cauwelaert, Peter Veelaert & Wilfried Philips. *Extrinsic Calibration of Camera Networks Using a Sphere*. MDPI Sensors, vol. 15, no. 8, pages 18985–19005, 2015.

[Gupta 1997] Rajiv Gupta & Richard I. Hartley. *Linear pushbroom cameras*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 19, no. 9, pages 963–975, 1997.

[Ha 2009] Jong-Eun Ha. *Automatic detection of chessboard and its applications*. Optical Engineering, vol. 48, no. 6, page 67205, 2009.

[Hafner 2014] David Hafner, Oliver Demetz, Joachim Weickert & Martin Reißel. *Mathematical Foundations and Generalisations of the Census Transform for Robust Optic Flow Computation*. Journal of Mathematical Imaging and Vision, pages 71–86, 2014.

[Hagen 2012] Nathan Hagen, Robert T. Kester, Liang Gao & Tomasz S. Tkaczyk. *Snapshot advantage: a review of the light collection improvement for parallel high-dimensional measurement systems*. Optical Engineering, vol. 51, no. 11, pages 111701–111702, 2012.

[Han 1995] Jun Han & Claudio Moraga. *The influence of the sigmoid function parameters on the speed of backpropagation learning*. From Natural to Artificial Neural Computation, vol. 195-201, 1995.

[Häne 2012] Christian Häne, Christopher Zach, Bernhard Zeisl & Marc Pollefeys. *A Patch Prior for Dense 3D Reconstruction in Man-Made Environments*. In IEEE International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission, pages 563–570, oct 2012.

[Harris 1988] Chris Harris & Mike Stephens. *A combined corner and edge detector.* In Alvey Vision Conference, volume 15, page 50, 1988.

[Hartley 1993] Richard I. Hartley. *Cheirality invariants*. In DARPA Image Understanding Workshop, pages 745–753, 1993.

[Hartley 1997] Richard I. Hartley & Peter F. Sturm. *Triangulation*. Computer Vision and Image Understanding, vol. 68, no. 2, pages 146–157, 1997.

[Hartley 2004] Richard I. Hartley & Andrew Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, second edition, 2004.

[Hartley 2008] Richard I. Hartley & René Vidal. *Perspective Nonrigid Shape and Motion Recovery*. In IEEE European Conference on Computer Vision (ECCV), pages 276–289, oct 2008.

[Hartley 2013] Richard I. Hartley, Fredrik Kahl, Carl Olsson & Yongduek Seo. *Verifying Global Minima for L2 Minimization Problems in Multiple View Geometry*. International Journal of Computer Vision (IJCV), vol. 101, no. 2, pages 288–304, 2013.

[Heber 2013] Stefan Heber, Rene Ranftl & Thomas Pock. *Variational Shape from Light Field*. In Energy Minimization Methods in Computer Vision and Pattern Recognition, volume 8081 of *Lecture Notes in Computer Science*, pages 66–79, 2013.

[Heikkila 1997] Janne Heikkila & Olli Silven. *A Four-step Camera Calibration Procedure with Implicit Image Correction*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 1997.

[Heinrich 2013] Mattias P. Heinrich, Mark Jenkinson, Bartlomiej W. Papież, Fergus V. Glesson, Sir Michael Brady & Julia A. Schnabel. *Edge- and Detail-preserving Sparse Image Representations for Deformable Registration of Chest MRI and CT Volumes*. In International Conference on Information Processing in Medical Imaging, pages 463–474, 2013.

[Hirschmüller 2005] Heiko Hirschmüller. *Accurate and Efficient Stereo Processing by Semi-Global Matching and Mutual Information*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 807–814, 2005.

[Hirschmüller 2007] Heiko Hirschmüller & Daniel Scharstein. *Evaluation of Cost Functions for Stereo Matching*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, jun 2007.

[Hirschmüller 2008] Heiko Hirschmüller. *Stereo Processing by Semiglobal Matching and Mutual Information*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 30, no. 2, pages 328–341, feb 2008.

[Huang 2017] Gao Huang, Zhuang Liu, Laurens van der Maaten & Kilian Q Weinberger. *Densely Connected Convolutional Networks*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017.

[Huber 2011] Peter J. Huber. *Robust Statistics*. International Encyclopedia of Statistical Science, pages 1248—-1251, 2011.

[Hui 2013] Bingwei Hui, Gongjian Wen, Peng Zhang & Deren Li. *A novel line scan camera calibration technique with an auxiliary frame camera*. IEEE Transactions on Instrumentation and Measurement, vol. 62, no. 9, pages 2567–2575, 2013.

[Ilg 2016] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy & Thomas Brox. *FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[Izadi 2011] Shahram Izadi, Andrew Davison, Andrew Fitzgibbon, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges & Dustin Freeman. *KinectFusion*. In ACM Symposium on User Interface Software and Technology (UST), page 559. ACM Press, 2011.

[Jeon 2015] Hae-Gon Jeon, Jaesik Park, Gyeongmin Choe, Jinsun Park, Yunsu Bok, Yu-Wing Tai & In So Kweon. *Accurate depth map estimation from a lenslet light field camera*. In 2IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1547–1555, 2015.

[Ji 2017] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu & Lu Fang. *SurfaceNet: An End-to-end 3D Neural Network for Multiview Stereopsis*. In IEEE International Conference on Computer Vision (ICCV), aug 2017.

[Jing Xiao 2005] Jing Xiao & Takeo Kanade. *Uncalibrated perspective reconstruction of deformable structures*. In IEEE Inter-

national Conference on Computer Vision (ICCV), pages 1075–1082, 2005.

[Kahl 2008a] Fredrik Kahl, Sameer Agarwal, ManmohanKrishna Chandraker, David Kriegman & Serge Belongie. *Practical Global Optimization for Multiview Geometry*. International Journal of Computer Vision (IJCV), vol. 79, no. 3, pages 271–284, 2008.

[Kahl 2008b] Fredrik Kahl & Richard I. Hartley. *Multiple-View Geometry Under the L-infinity Norm*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 30, no. 9, pages 1603–1617, sep 2008.

[Kannala 2006] Juho Kannala & Sami S. Brandt. *A generic camera model and calibration method for conventional, wide-angle, and fish-eye lenses*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 28, no. 8, pages 1335–1340, 2006.

[Kar 2017] Abhishek Kar, Christian Häne & Jitendra Malik. Learning a Multi-View Stereo Machine. aug 2017.

[Kendall 2017] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach & Adam Bry. *End-to-End Learning of Geometry and Context for Deep Stereo Regression*. In IEEE International Conference on Computer Vision (ICCV), mar 2017.

[Kennedy 2015] Ryan Kennedy & Camillo J. Taylor. *Optical flow with geometric occlusion estimation and fusion of multiple frames*. In Energy Minimization Methods in Computer Vision and Pattern Recognition, pages 364–377, 2015.

[Kerl 2013] Christian Kerl, Jurgen Sturm & Daniel Cremers. *Dense visual SLAM for RGB-D cameras*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 2100–2106, nov 2013.

[Kim 2013] Changil Kim, Henning Zimmer, Yael Pritch, Alexander Sorkine-Hornung & Markus Gross. *Scene Reconstruction from High Spatio-angular Resolution Light Fields*. ACM Transactions on Graphics, vol. 32, no. 4, 2013.

[Kimmel 1999] Ron Kimmel. *Demosaicing: image reconstruction from color CCD samples*. IEEE Transactions on Image Processing, vol. 8, no. 9, pages 1221–1228, 1999.

[Klein 2007] Georg Klein & David Murray. *Parallel tracking and mapping for small AR workspaces*. In IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR), pages 1–10, nov 2007.

[Kopf 2013] Johannes Kopf, Ariel Shamir & Pieter Peers. *Content-adaptive Image Downscaling*. ACM Transactions on Graphics, vol. 32, no. 6, nov 2013.

[Kummerle 2011] Rainer Kummerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige & Wolfram Burgard. *G2O: A general framework for graph optimization*. In IEEE International Conference on Robotics and Automation (ICRA), pages 3607–3613, may 2011.

[Kuschk 2013] Georg Kuschk & Daniel Cremers. *Fast and Accurate Large-Scale Stereo Reconstruction Using Variational Methods*. In IEEE International Conference on Computer Vision (ICCV), pages 700–707, dec 2013.

[Kutulakos 2000] Kiriakos N Kutulakos & Steven M Seitz. *A theory of shape by space carving*. International Journal of Computer Vision (IJCV), vol. 38, no. 3, pages 199–218, 2000.

[Kılıç 2008] Emrah Kılıç. *Explicit formula for the inverse of a tridiagonal matrix by backward continued fractions*. Applied Mathematics and Computation, vol. 197, no. 1, pages 345–357, 2008.

[Labatut 2007] Patrick Labatut, Jean-Philippe Pons & Renaud Keriven. *Efficient multi-view reconstruction of large-scale scenes using interest points, delaunay triangulation and graph cuts*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2007.

[Laurentini 1994] Aldo Laurentini. *The visual hull concept for silhouette-based image understanding*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 16, no. 2, pages 150–162, 1994.

[LeCun 2015] Yann LeCun, Yoshua Bengio & Geoffrey Hinton. *Deep learning*. Nature2, vol. 521, no. 7553, pages 436–444, 2015.

[Lee 2011] Sung Joo Lee, Kang Ryoung Park & Jaihie Kim. *A SfM-based 3D face reconstruction method robust to self-occlusion by using a shape conversion matrix*. Pattern Recognition, vol. 44, no. 7, pages 1470–1486, jul 2011.

[Lehtomäki 2010] Matti Lehtomäki, Anttoni Jaakkola, Juha Hyyppä, Antero Kukko & Harri Kaartinen. *Detection of vertical pole-like objects in a road environment using vehicle-based laser scanning data*. MDPI Remote Sensing, vol. 2, no. 3, pages 641–664, 2010.

[Levenberg 1944] Kenneth Levenberg. *A method for the solution of certain non-linear problems in least squares*. Quarterly of Applied Mathematics, vol. 2, no. 2, pages 164–168, 1944.

[Levinshtein 2009] Alex Levinshtein, Adrian Stere, Kiriakos N. Kutulakos, David J. Fleet, Sven J. Dickinson & Kaleem Siddiqi. *TurboPixels: Fast Superpixels Using Geometric Flows*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 31, no. 12, dec 2009.

[Levoy 1996] Marc Levoy & Pat Hanrahan. *Light Field Rendering*. In Special Interest Group on Computer Graphics and Interactive Techniques (SIGGRAPH), pages 31–42, New York, NY, USA, 1996.

[Li 2008] Xin Li, Bahadir Gunturk & Lei Zhang. *Image demosaicing: A systematic survey*. In Electronic Imaging, 2008.

[Li 2013] Dong Li & Jindong Tian. *An accurate calibration method for a camera with telecentric lenses*. Optics and Lasers in Engineering, vol. 52, no. 5, pages 538–541, 2013.

[Li 2016] Dongdong Li, Gongjian Wen & Shaohua Qiu. *Cross-ratio–based line scan camera calibration using a planar pattern*. Optical Engineering, vol. 55, no. 1, 2016.

[Liao 2013] Wenzhi Liao, Bart Goossens, Jan Aelterman, Hiep Luong, Aleksandra Pizurica, Niels Wouters, Wouter Saeys & Wilfried Philips. *Hyperspectral image deblurring with PCA*

and total variation. In IEEE GRSS Workshop Hyperspectral Image SIgnal Processing: Evolution in Remote Sensing (WHISPERS), pages 1–4, 2013.

[Loop 1999] Charles Loop & Zhengyou Zhang. *Computing rectifying homographies for stereo vision*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 1, pages 125–131, 1999.

[Lord Rayleigh 1891] Lord Rayleigh. *Some applications of photography*. Nature, vol. 44, no. 1133, pages 249–254, 1891.

[Lourakis 2009] Manolis I. A. Lourakis & Antonis A. Argyros. *SBA: A Software Package for Generic Sparse Bundle Adjustment*. ACM Transactions on Mathematical Software, vol. 36, no. 1, pages 1–30, 2009.

[Lowe 1999] David G. Lowe. *Object recognition from local scale-invariant features*. In IEEE International Conference on Computer Vision (ICCV), pages 1150–1157, 1999.

[Lu 2014] Guolan Lu & Baowei Fei. *Medical hyperspectral imaging: a review*. Journal of Biomedical Optics, vol. 19, no. 1, 2014.

[Lu 2015] Shao-Ping Lu, Beerend Ceulemans, Adrian Munteanu & Peter Schelkens. *Spatio-temporally consistent color and structure optimization for multiview video color correction*. IEEE Transactions on Multimedia, vol. 17, no. 5, pages 577–590, 2015.

[Lumsdaine 2009] Andrew Lumsdaine & Todor Georgiev. *The focused plenoptic camera*. In International Conference on Computational Photography (ICCP), pages 1—-8, 2009.

[Luo 2016] Wenjie Luo, Alexander G. Schwing & Raquel Urtasun. *Efficient Deep Learning for Stereo Matching*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5695–5703, jun 2016.

[Mallik 2001] Ranjan K. Mallik. *The inverse of a tridiagonal matrix*. Linear Algebra and its Applications, vol. 325, no. 1, pages 109–139, 2001.

[Mallon 2007] John Mallon & Paul F. Whelan. *Calibration and removal of lateral chromatic aberration in images*. Pattern Recognition Letters, vol. 28, no. 1, pages 125–135, 2007.

[Marquardt 1963] Donald W. Marquardt. *An Algorithm for Least-Squares Estimation of Nonlinear Parameters*. Journal of the society for Industrial and Applied Mathematics, vol. 11, no. 2, pages 431–441, 1963.

[Mayer 2016] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy & Thomas Brox. *A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4040–4048, jun 2016.

[Moravec 1977] Hans P. Moravec. *Towards Automatic Visual Obstacle Avoidance*. In International Conference on Artificial Intelligence, 1977.

[Mori 2004] Greg Mori, Xiaofeng Ren, Alexei A. Efros & Jitendra Malik. *Recovering human body configurations: combining segmentation and recognition*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, 2004.

[Mori 2005] Greg Mori. *Guiding model search using segmentation*. In IEEE International Conference on Computer Vision (ICCV), volume 2, 2005.

[Mur-Artal 2015] Raul Mur-Artal, Jose Maria Martinez Montiel & Juan D. Tardos. *ORB-SLAM: A Versatile and Accurate Monocular SLAM System*. IEEE Transactions on Robotics, vol. 31, no. 5, pages 1147–1163, oct 2015.

[Nair 2010] Vinod Nair & Geoffrey E. Hinton. *Rectified Linear Units Improve Restricted Boltzmann Machines*. In International Conference on Machine Learning (ICML), pages 807–814, 2010.

[Ng 2005] Ren Ng, Marc Levoy, Gene Duval, Mark Horowitz & Pat Hanrahan. *Light Field Photography with a Hand-held Plenoptic Camera*. Technical report, 2005.

[Nga 2013] Do Hang Nga & Keiji Yanai. *A Spatio-temporal Feature Based on Triangulation of Dense SURF*. In IEEE International Conference on Computer Vision (ICCV), pages 420–427, dec 2013.

[Ngiam 2011] Jiquan Ngiam, Adam Coates, Ahbik Lahiri, Bobby Prochnow, Quoc V. Le & Andrew Y. Ng. *On optimization methods for deep learning*. In International Conference on Machine Learning (ICML), pages 265–272, 2011.

[Nießner 2013] Matthias Nießner, Michael Zollhöfer & Shahram Izadi. *Real-time 3D reconstruction at scale using voxel hashing*. ACM Transactions on Graphics, vol. 32, no. 6, 2013.

[Nishigaki 2008] Morimichi Nishigaki. *Color Segmentation-based Optical Flow Computation and Motion Segmentation*. Technical report, Department of Computer Science, University of Maryland, 2008.

[Olsson 2007] Carl Olsson, Anders P. Eriksson & Fredrik Kahl. *Efficient optimization for L∞-problems using pseudoconvexity*. In IEEE International Conference on Computer Vision (ICCV), oct 2007.

[Olsson 2010] Carl Olsson, Anders Eriksson & Richard Hartley. *Outlier removal using duality*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1450–1457. IEEE, 2010.

[Olsson 2011] Carl Olsson & Olof Enqvist. *Stable Structure from Motion for Unordered Image Collections*. Image Analysis, vol. 6688, pages 524–535, 2011.

[Oren 1995] Michael Oren & Shree K Nayar. *Generalization of the Lambertian model and implications for machine vision*. International Journal of Computer Vision (IJCV), vol. 14, no. 3, pages 227–251, 1995.

[Paladini 2010] Marco Paladini, Adrien Bartoli & Lourdes Agapito. *Sequential Non-Rigid Structure-from-Motion with the 3D-Implicit Low-Rank Shape Model*. In IEEE European Conference on Computer Vision (ECCV), pages 15–28, 2010.

[Paysan 2009] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani & Thomas Vetter. *A 3D Face Model for Pose*

and Illumination Invariant Face Recognition. In IEEE International Conference on Advanced Video and Signal Based Surveillance, pages 296–301, sep 2009.

[Perwass 2012] Christian Perwass & Lennart Wietzke. *Single lens 3D-camera with extended depth-of-field*. Human Vision and Electronic Imaging, vol. 8291, 2012.

[Placht 2014] Simon Placht, Peter Fürsattel, Etienne Assoumou Mengue, Hannes Hofmann, Christian Schaller, Michael Balda & Elli Angelopoulou. *Rochade: Robust checkerboard advanced detection for camera calibration*. In IEEE European Conference on Computer Vision (ECCV), pages 766–779, 2014.

[Pollefeys 2008] Marc Pollefeys, David Nistér, J-M Frahm, Amir Akbarzadeh, Philippos Mordohai, Brian Clipp, Chris Engels, David Gallup, S-J Kim & Paul Merrell. *Detailed real-time urban 3d reconstruction from video*. International Journal of Computer Vision (IJCV), vol. 78, no. 2, pages 143–167, 2008.

[Powell 2008] Stephanie Powell, Vincent A Magnotta, Hans Johnson, Vamsi K. Jammalamadaka, Ronald Pierson & Nancy C. Andreasen. *Registration and machine learning-based automated segmentation of subcortical and cerebellar brain structures*. Neuroimage, vol. 39, no. 1, pages 238–247, 2008.

[Rabaud 2008] Vincent Rabaud & Serge Belongie. *Rethinking Non-rigid Structure-from-motion*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.

[Ranftl 2012] Rene Ranftl, Stefan Gehrig, Thomas Pock & Horst Bischof. *Pushing the Limits of Stereo Using Variational Stereo Estimation*. In IEEE Intelligent Vehicles Symposium (IV), 2012.

[Ranftl 2013] Rene Ranftl, Thomas Pock & Horst Bischof. *Minimizing TGV-Based Variational Models with Non-convex Data Terms*. In International Conference on Scale Space and Variational Methods in Computer Vision, pages 282–293, 2013.

[Ranjan 2016] Anurag Ranjan & Michael J. Black. *Optical Flow Estimation using a Spatial Pyramid Network*. In arXiv preprint, 2016.

[Ren 2003] Xiaofeng Ren & Jitendra Malik. *Learning a Classification Model for Segmentation*. In IEEE International Conference on Computer Vision (ICCV), 2003.

[Ren 2011] Carl Yuheng Ren & Ian Reid. *gSLIC: a real-time implementation of SLIC superpixel segmentation*. Technical report, University of Oxford, Department of Engineering Science, 2011.

[Rockafellar 1976] R. Tyrrell Rockafellar. *Monotone Operators and the Proximal Point Algorithm*. SIAM Journal on Control and Optimization, vol. 14, no. 5, pages 877–898, 1976.

[Roels 2016] Joris Roels, Jonas De Vylder, Jan Aelterman, Yvan Saeys & Wilfried Philips. *Automated Membrane Detection in Electron Microscopy using Convolutional Neural Networks*. In Belgian-Dutch Conference on Machine Learning (BENELEARN), 2016.

[Rosten 2006] Edward Rosten & Tom Drummond. *Machine learning for high-speed corner detection*. In IEEE European Conference on Computer Vision (ECCV), pages 430–443, 2006.

[Rosten 2010] Edward Rosten, Reid Porter & Tom Drummond. *FASTER and better: A machine learning approach to corner detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 32, pages 105–119, 2010.

[Rublee 2011] Ethan Rublee, Vincent Rabaud, Kurt Konolige & Gary Bradski. *ORB: an efficient alternative to SIFT or SURF*. In IEEE International Conference on Computer Vision (ICCV), pages 2564–2571, 2011.

[Rufli 2008] Martin Rufli, Davide Scaramuzza & Roland Siegwart. *Automatic detection of checkerboards on blurred and distorted images*. In IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 3121–3126, sep 2008.

[Salas-Moreno 2014] Renato F. Salas-Moreno, Ben Glocken, Paul H. J. Kelly & Andrew J. Davison. *Dense planar SLAM*. In IEEE International Symposium on Mixed and Augmented Reality (ISMAR), pages 157–164. IEEE, sep 2014.

[Scharstein 2002] Dabiel Scharstein & Richard Szeliski. *A taxonomy and evaluation of dense two-frame stereo correspondence algorithms*. International Journal of Computer Vision (IJCV), pages 131–140, 2002.

[Scharstein 2003] Daniel Scharstein & Richard Szeliski. *High-Accuracy Stereo Depth Maps Using Structured Light*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2003.

[Scharstein 2007] Daniel Scharstein & Chris Pal. *Learning Conditional Random Fields for Stereo*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2007.

[Scharstein 2014] Daniel Scharstein, Heiko Hirschmüller, York Kitajima, Greg Krathwohl, Nera Nesic, Xi Wang & Porter Westling. *High-Resolution Stereo Datasets with Subpixel-Accurate Ground Truth*. In German Conference on Pattern Recognition (GCPR), 2014.

[Schmidhuber 2015] Jürgen Schmidhuber. *Deep learning in neural networks: An overview*. Neural networks, vol. 61, pages 85–117, 2015.

[Schönberger 2016a] Johannes Lutz Schönberger & Jan-Michael Frahm. *Structure-from-Motion Revisited*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.

[Schönberger 2016b] Johannes Lutz Schönberger, Enliang Zheng, Marc Pollefeys & Jan-Michael Frahm. *Pixelwise View Selection for Unstructured Multi-View Stereo*. In IEEE European Conference on Computer Vision (ECCV), 2016.

[Seki 2016] Akihito Seki & Marc Pollefeys. *Patch Based Confidence Prediction for Dense Disparity Map*. In British Machine Vision Conference (BMVC), pages 23.1–23.13, 2016.

[Seki 2017] Akihito Seki & Marc Pollefeys. *SGM-Nets: Semi-Global Matching with Neural Networks*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6640–6649. IEEE, jul 2017.

[Shaji 2008] Appu Shaji & Sharat Chandran. *Riemannian manifold optimisation for non-rigid structure from motion*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–6. IEEE, jun 2008.

[Shan 2014] Qi Shan, Brian Curless, Yasutaka Furukawa, Carlos Hernandez & Steven M. Seitz. *Occluding contours for multi-view stereo*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 4002–4009. IEEE, jun 2014.

[Smith 1997] Stephen M. Smith & J. Michael Brady. *SUSAN—a new approach to low level image processing*. International Journal of Computer Vision (IJCV), vol. 23, no. 1, pages 45–78, 1997.

[Staunton 1989] Richard Staunton. *Hexagonal image sampling: A practical proposition*. In Expert Robots for Industrial Use, pages 23–28, 1989.

[Steger 1998] Carsten Steger. *An unbiased detector of curvilinear structures*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 20, no. 2, pages 113–125, 1998.

[Steinbrücker 2014] Frank Steinbrücker, Jürgen Sturm & Daniel Cremers. *Volumetric 3D mapping in real-time on a CPU*. In IEEE International Conference on Robotics and Automation (ICRA), pages 2021–2028, 2014.

[Stewenius 2005] Henrik Stewenius, Frederik Schaffalitzky & David Nister. *How hard is 3-view triangulation really?* In IEEE International Conference on Computer Vision (ICCV), volume 1, pages 686–693, oct 2005.

[Sturm 1999] Jurgen Sturm. *Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones*. Optimization Methods and Software, vol. 11–12, pages 625–653, 1999.

[Sturm 2011] Peter Sturm, Srikumar Ramalingam, Jean-Philippe Tardif, Simone Gasparini & Joao Barreto. *Camera models and fundamental concepts used in geometric computer vision*. Foundations and Trends in Computer Graphics and Vision, vol. 6, no. 1–2, pages 1–183, 2011.

[Su 2013] Jiandong Su, Xiusheng Duan & Jing Xiao. *Fast detection method of checkerboard corners based on the combination of template matching and Harris Operator*. In International Conference on Information Science and Technology (ICIST), pages 858–861. IEEE, 2013.

[Sun 2016] Bo Sun, Jigui Zhu, Linghui Yang, Shourui Yang & Zhiyuan Niu. *Calibration of line-scan cameras for precision measurement*. Applied Optics, vol. 55, no. 25, pages 6836–6843, 2016.

[Sutskever 2013] Ilya Sutskever, James Martens, George Dahl & Geoffrey Hinton. *On the importance of initialization and momentum in deep learning*. In International Conference on Machine Learning (ICML), pages 1139–1147, 2013.

[Svensson 2002] Stina Svensson, Ingela Nyström & G Sanniti Di Baja. *Curve skeletonization of surface-like objects in 3D images guided by voxel classification*. Pattern Recognition Letters, vol. 23, no. 12, pages 1419–1426, 2002.

[Tao 2013] Lili Tao, Stephen J. Mein, Wei Quan & Bogdan J. Matuszewski. *Recursive non-rigid structure from motion with online learned shape prior*. Computer Vision and Image Understanding, vol. 117, no. 10, pages 1287–1298, oct 2013.

[Thormählen 2006] Thorsten Thormählen, Hellward Broszio & Patrick Mikulastik. *Robust linear auto-calibration of a moving camera from image sequences*. In IEEE Asian Conference on Computer Vision (ACCV), pages 71–80, 2006.

[Thormählen 2012] Thorsten Thormählen & Hellward Broszio. *VOODOO camera tracker*. http://www.viscoda.com/, 2012.

[Tomasi 1992] Carlo Tomasi & Takeo Kanade. *Shape and motion from image streams under orthography: a factorization method*. International Journal of Computer Vision (IJCV), vol. 9, no. 2, pages 137–154, 1992.

[Torresani 2008] Lorenzo Torresani, Aaron Hertzmann & Chris Bregler. *Nonrigid structure-from-motion: Estimating shape and motion with hierarchical priors*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 30, no. 5, pages 878–892, 2008.

[Tosic 2014] Ivana Tosic & Kathrin Berkner. *Light field scale-depth space transform for dense depth estimation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 435–442, 2014.

[Triggs 1999] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley & Andrew W. Fitzgibbon. *Bundle Adjustment — A Modern Synthesis*. In International Workshop on Vision Algorithms, pages 298–372, 1999.

[Van den Bergh 2012] Michael Van den Bergh & Luc Van Gool. *Real-time stereo and flow-based video segmentation with superpixels*. In IEEE Workshop on Applications of Computer Vision (WACV), pages 89–96, jan 2012.

[Vedaldi 2008] Andrea Vedaldi & Stefano Soatto. *Quick Shift and Kernel Methods for Mode Seeking*. In IEEE European Conference on Computer Vision (ECCV), volume 5305 of *Lecture Notes in Computer Science*, 2008.

[Vezhnevets 2005] Vladimir Vezhnevets. *OpenCV Calibration Object Detection*, 2005.

[Vlaminck 2017] Michiel Vlaminck, Hiep Luong & Wilfried Philips. *A Markerless 3D Tracking Approach for Augmented Reality Applications*. In IEEE International Conference on 3D Immersion (IC3D), 2017.

[Wang 2007] Zhongshi Wang, Wei Wu, Xinhe Xu & Dingyu Xue. *Recognition and location of the internal corners of planar checkerboard calibration pattern image*. Applied Mathematics and Computation, vol. 185, no. 2, pages 894–906, 2007.

[Wang 2008] Shu-Fan Wang & Shang-Hong Lai. *Estimating 3D Face Model and Facial Deformation from a Single Image Based on Expression Manifold Optimization*. In IEEE European Conference on Computer Vision (ECCV), pages 589–602. Springer, Berlin, Heidelberg, oct 2008.

[Wang 2015] Ting-Chun Wang, Alexei A. Efros & Ravi Ramamoorthi. *Occlusion-aware depth estimation using light-field cameras*. In IEEE International Conference on Computer Vision (ICCV), pages 3487–3495, 2015.

[Wang 2017] Rui Wang, Martin Schwörer & Daniel Cremers. *Stereo DSO: Large-Scale Direct Sparse Visual Odometry with Stereo Cameras*. IEEE International Conference on Computer Vision (ICCV), aug 2017.

[Wanner 2013] Sven Wanner, Stephan Meister & Bastian Goldluecke. *Datasets and Benchmarks for Densely Sampled 4D Light Fields*. In Vision, Modeling & Visualization (VMV), pages 225–226, 2013.

[Webb 1982] Jon A. Webb & Jake K. Aggarwal. *Structure from motion of rigid and jointed objects*. Artificial Intelligence, vol. 19, no. 1, pages 107–130, sep 1982.

[Wu 2005] F. C. Wu, Z. Y. Hu & H. J. Zhu. *Camera calibration with moving one-dimensional objects*. Pattern Recognition, vol. 38, no. 5, pages 755–765, 2005.

[Xu 2012] Li Xu, Jiaya Jia & Yasuyuki Matsushita. *Motion Detail Preserving Optical Flow Estimation*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 34, no. 9, pages 1744–1757, 2012.

[Ye 2017] Siqi Ye, Shao-Ping Lu & Adrian Munteanu. *Color correction for large-baseline multiview video*. Signal Processing: Image Communication, vol. 53, pages 40–50, 2017.

[Zabih 1994] Ramin Zabih & John Woodfill. *Non-parametric local transforms for computing visual correspondence*. In IEEE European Conference on Computer Vision (ECCV), pages 151–158, 1994.

[Žbontar 2016] Jure Žbontar & Yann LeCun. *Stereo Matching by Training a Convolutional Neural Network to Compare Image Patches*. Journal of Machine Learning Research, vol. 17, no. 65, pages 1–32, 2016.

[Zhang 1999] Zhengyou Zhang. *Flexible Camera Calibration by Viewing a Plane from Unknown Orientations*. In IEEE International Conference on Computer Vision (ICCV), 1999.

[Zhang 2000] Zhengyou Zhang. *A flexible new technique for camera calibration*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 22, no. 11, pages 1330–1334, 2000.

[Zhang 2004] Zhengyou Zhang. *Camera calibration with one-dimensional objects*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 26, no. 7, pages 892–899, 2004.

[Zhang 2007] Hui Zhang, K. Wong Kwan-yee & Guoqiang Zhang. *Camera calibration from images of spheres*. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), vol. 29, no. 3, pages 499–502, 2007.

[Zhang 2008] Hui Zhang, Jason E. Fritts & Sally A. Goldman. *Image segmentation evaluation: A survey of unsupervised methods*. Computer Vision and Image Understanding, vol. 110, no. 2, pages 260–280, 2008.

[Zhang 2009] Ke Zhang, Jiangbo Lu, Gauthier Lafruit, Rudy Lauwereins & Luc Van Gool. *Robust stereo matching with fast Normalized Cross-Correlation over shape-adaptive regions*. In IEEE International Conference on Image Processing (ICIP), pages 2357–2360. IEEE, nov 2009.

[Zhang 2010] Cha Zhang & Zhengyou Zhang. *A Survey of Recent Advances in Face Detection*. Technical report June, Microsoft Research, 2010.

[Zheng 2014] Enliang Zheng, Enrique Dunn, Vladimir Jojic & Jan Michael Frahm. *PatchMatch based joint view selection and depthmap estimation*. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1510–1517. IEEE, jun 2014.

[Zhou 2012] Huiyu Zhou, Xuelong Li & Abdul H. Sadka. *Nonrigid structure-from-motion from 2-d images using markov chain monte carlo*. IEEE Transactions on Multimedia, vol. 14, no. 1, pages 168–177, 2012.

[Zhu 2009] Weixing Zhu, Changhua Ma, Libing Xia & Xincheng Li. *A fast and accurate algorithm for chessboard corner detection*. In International Congress on Image and Signal Processing (CISP), pages 1–5, 2009.