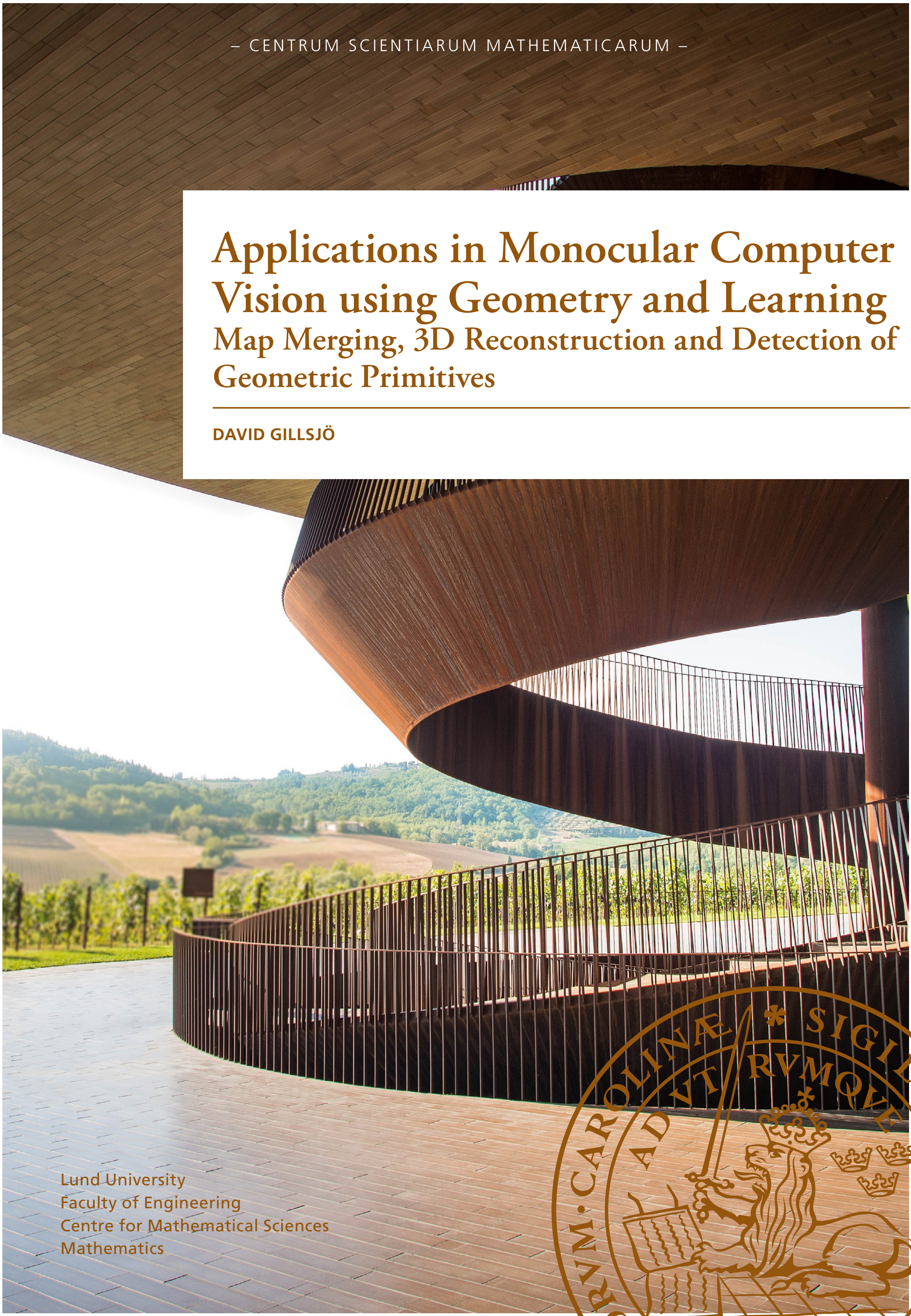
– CENTRUM SCIENTIARUM MATHEMATICARUM –



Applications in Monocular Computer Vision using Geometry and Learning
Map Merging, 3D Reconstruction and Detection of Geometric Primitives

DAVID GILLSJÖ

Lund University
Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Applications in Monocular Computer Vision using Geometry and Learning

Map Merging, 3D Reconstruction and Detection of Geometric Primitives

by David Gillsjö

LUND UNIVERSITY

ACADEMIC THESIS

which, with due permission of the Faculty of Engineering at Lund University, will be publicly defended on Friday 2nd of June, 2023, at 13:15 in lecture hall MH:Hörmandersalen, Centre for Mathematical Sciences, Sölvegatan 18, Lund for the degree of Doctor of Philosophy in Engineering.

Thesis advisors:
Prof. Kalle Åström, Prof. Anders Heyden, Dr. Gabrielle Flood

Faculty opponent:
Prof. Atsuto Maki, KTH, Sweden

Organization **LUND UNIVERSITY** Centre for Mathematical Sciences Box 118 SE–221 00 LUND Sweden	Document name **Doctoral thesis**
	Date of presentation 2023-06-02
Author(s) David Gillsjö	Sponsoring organization WASP, SSF, ELLIT

Title and subtitle

Applications in Monocular Computer Vision using Geometry and Learning – Map Merging, 3D Reconstruction and Detection of Geometric Primitives

Abstract

As the dream of autonomous vehicles moving around in our world comes closer, the problem of robust localization and mapping is essential to solve. In this inherently structured and geometric problem we also want the agents to learn from experience in a data driven fashion. How the modern Neural Network models can be combined with Structure from Motion (SfM) is an interesting research question and this thesis studies some related problems in 3D reconstruction, feature detection, SfM and map merging.

In Paper I we study how a Bayesian Neural Network (BNN) performs in Semantic Scene Completion, where the task is to predict a semantic 3D voxel grid for the Field of View of a single RGBD image. We propose an extended task and evaluate the benefits of the BNN when encountering new classes at inference time. It is shown that the BNN outperforms the deterministic baseline.

Papers II-III are about detection of points, lines and planes defining a Room Layout in an RGB image. Due to the repeated textures and homogeneous colours of indoor surfaces it is not ideal to only use point features for Structure from Motion. The idea is to complement the point features by detecting a Wireframe – a connected set of line segments – which marks the intersection of planes in the Room Layout. Paper II concerns a task for detecting a Semantic Room Wireframe and implements a Neural Network model utilizing a Graph Convolutional Network module. The experiments show that the method is more flexible than previous Room Layout Estimation methods and perform better than previous Wireframe Parsing methods. Paper III takes the task closer to Room Layout Estimation by detecting a connected set of semantic polygons in an RGB image. The end-to-end trainable model is a combination of a Wireframe Parsing model and a Heterogeneous Graph Neural Network. We show promising results by outperforming state of the art models for Room Layout Estimation using synthetic Wireframe detections. However, the joint Wireframe and Polygon detector requires further research to compete with the state of the art models.

In Paper IV we propose minimal solvers for SfM with parallel cylinders. The problem may be reduced to estimating circles in 2D and the paper contributes with theory for the two-view relative motion and two-circle relative structure problem. Fast solvers are derived and experiments show good performance in both simulation and on real data.

Papers V-VII cover the task of map merging. That is, given a set of individually optimized point clouds with camera poses from a SfM pipeline, how can the solutions be effectively merged without completely re-solving the Structure from Motion problem? Papers V-VI introduce an effective method for merging and shows the effectiveness through experiments of real and simulated data. Paper VII considers the matching problem for point clouds and proposes minimal solvers that allows for deformation of each point cloud. Experiments show that the method robustly matches point clouds with drift in the SfM solution.

Key words

Scene Completion ; Deep Learning ; Bayesian Neural Network ; Room Layout Estimation ;
Line Segment Detection; Polygon Detection ; Graph Neural Network ; Map Merging ; Structure from Motion ;
Minimal Solvers ; Wireframe Estimation ; Sampson Approximation

Classification system and/or index terms (if any)

Supplementary bibliographical information		Language English
ISSN and key title 1404-0034		ISBN 978-91-8039-643-1 (print) 978-91-8039-644-8 (electronic)
Recipient's notes	Number of pages xxii+216	Price
	Security classification	

Signature   Date 2023-04-18

Applications in Monocular Computer Vision using Geometry and Learning

Map Merging, 3D Reconstruction and Detection of Geometric Primitives

by David Gillsjö

LUND UNIVERSITY

Centre for Mathematical Sciences
Lund University
Box 118
SE-221 00 Lund
Sweden
www.maths.lu.se

Doctoral Theses in Mathematical Sciences 2023:2
ISSN: 1404-0034
ISBN: 978-91-8039-643-1 (print)
ISBN: 978-91-8039-644-8 (electronic)
LUTFMA-1081-2023

Printed in Sweden by Media-Tryck, Lund University, Lund 2023

If we knew what it was we were doing,
it would not be called research,
would it?

Albert Einstein

Abstract

As the dream of autonomous vehicles moving around in our world comes closer, the problem of robust localization and mapping is essential to solve. In this inherently structured and geometric problem we also want the agents to learn from experience in a data driven fashion. How the modern Neural Network models can be combined with Structure from Motion (SfM) is an interesting research question and this thesis studies some related problems in 3D reconstruction, feature detection, SfM and map merging.

In Paper I we study how a Bayesian Neural Network (BNN) performs in Semantic Scene Completion, where the task is to predict a semantic 3D voxel grid for the Field of View of a single RGBD image. We propose an extended task and evaluate the benefits of the BNN when encountering new classes at inference time. It is shown that the BNN outperforms the deterministic baseline.

Papers II-III are about detection of points, lines and planes defining a Room Layout in an RGB image. Due to the repeated textures and homogeneous colours of indoor surfaces it is not ideal to only use point features for Structure from Motion. The idea is to complement the point features by detecting a Wireframe – a connected set of line segments – which marks the intersection of planes in the Room Layout. Paper II concerns a task for detecting a Semantic Room Wireframe and implements a Neural Network model utilizing a Graph Convolutional Network module. The experiments show that the method is more flexible than previous Room Layout Estimation methods and perform better than previous Wireframe Parsing methods. Paper III takes the task closer to Room Layout Estimation by detecting a connected set of semantic polygons in an RGB image. The end-to-end trainable model is a combination of a Wireframe Parsing model and a Heterogeneous Graph Neural Network. We show promising results by outperforming state of the art models for Room Layout Estimation using synthetic Wireframe detections. However, the joint Wireframe and Polygon detector requires further research to compete with the state of the art models.

In Paper IV we propose minimal solvers for SfM with parallel cylinders. The problem may be reduced to estimating circles in 2D and the paper contributes with theory for the two-view relative motion and two-circle relative structure problem. Fast solvers are derived and experiments show good performance in both simulation and on real data.

Papers V-VII cover the task of map merging. That is, given a set of individually optimized point clouds with camera poses from a SfM pipeline, how can the solutions be effectively merged without completely re-solving the Structure from Motion problem? Papers V-VI introduce an effective method for merging and shows the effectiveness through experiments of real and simulated data. Paper VII considers the matching problem for point clouds and proposes minimal solvers that allows for deformation of each point cloud. Experiments show that the method robustly matches point clouds with drift in the SfM solution.

Popular Summary

As humans, we are capable of perceiving all three spatial dimensions (3D) of our surroundings through our eyes and motion alone. Even though our eyes – like cameras – only capture a flat projection of the actual world, we have learned how to think in 3D. For example, by moving around in a house we learn the layout and size of the rooms; by walking around a block in the city we can build a sense of direction; through experience we can estimate the height of a cabinet without relying on a measuring tape.

Can we teach a computer to do this? This thesis studies a set of related questions in the field of *Computer Vision*. As tools we use geometrical mathematical models and *neural networks*, which are complex mathematical models inspired by the human brain models that require a large amount of data to learn from. Examples of questions are: Can we train neural networks to reason about hidden spaces indoors, as in Figure 1? Can we train neural networks to figure out the room layout in terms of floors, walls and ceilings from a single photo? How can we efficiently harness motion to estimate the 3D world from photos taken at different locations?

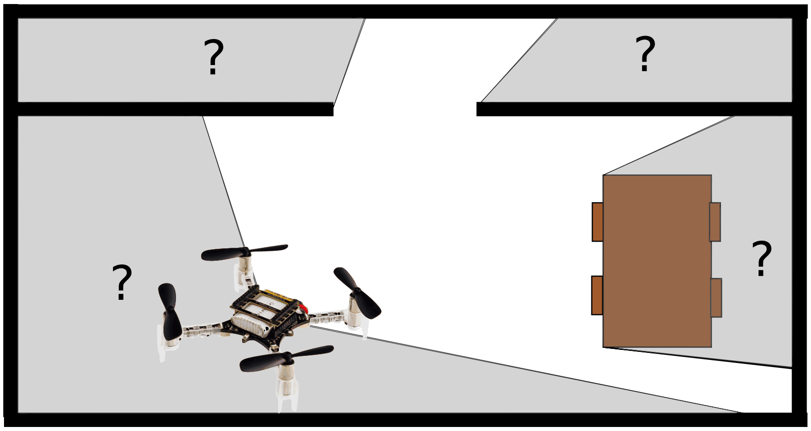


Figure 1: Quadcopter in a room seen from above. It is trying to guess what may be hidden from view behind the walls and table.

The brain's ability for abstract reasoning is one of many keys to our spatial awareness. To orient ourselves we may make note of key objects – such as houses, signs, trees etc. – that we use to track our current motion and help us find our way when we revisit the location. Within Computer Vision, the problem of understanding the 3D environment from photos is known as *Structure from Motion* (SfM). The first step is typically to identify good key objects or pixels in the photo that we can use to relate to the other photos.

The standard solution is to use *keypoints* in the image, with a colour intensity variation signature called a *descriptor*. The descriptor is used to find tentative matching points in other

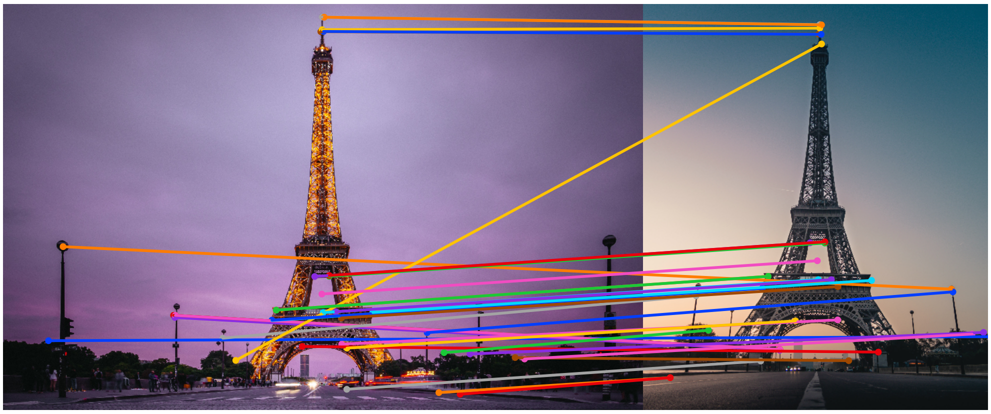


Figure 2: Two photos taken of the Eiffel Tower at different times from different locations. Keypoints (SIFT) are extracted from each image and matching by appearance is attempted. We see that while some matches are accurate, a large part of the tower has no matches and some are incorrect.
Image credit: Denys Nevozhai and Gautier Salles from `unsplash.com`.

images. While often sufficient, the matching methods struggles with repeated patterns like brick walls or homogeneous surfaces such as white painted walls. Changes over time are also difficult – for example, in Sweden a tree in winter looks very different to a tree in summer. In Figure 2 we see keypoints extracted from two images of the Eiffel Tower, but at different times of the day. There are only some parts of the tower that match correctly under these differing circumstances.

To alleviate these problems this thesis shows how e.g. trees and poles can be modelled as parallel cylinders to improve robustness and efficiency. Additionally it studies indoor scenes and how lines and polygons may be detected to represent the room layout in terms of floors, walls, ceilings, windows and doors as illustrated in Figure 3. This higher level of abstraction regarding key objects also improves robustness and is essential to reach human levels of 3D reasoning.

After key points and objects have been identified, it is time to estimate the pose (position and orientation) of the objects and of the cameras used to capture the photos. To find matching points or objects between image pairs, a robust matching method is needed that is capable of ignoring incorrect matches. A common method is the *Random Sample Consensus* (RANSAC) algorithm, which repeatedly solves a *minimal problem* for an alternating small set of random points. It is important to develop fast solvers for these minimal problems, as this makes it possible to match images in real time or in large scale. For example, estimating relative position and orientation between two calibrated cameras requires at least 5 corresponding points in each image. The solvers for this problem developed by the research community requires only microseconds to execute. Therefore, it is possible to perform thousands of RANSAC iterations in only milliseconds to find the best possible set of points. In this work, fast solvers are presented for both matching of parallel cylinders

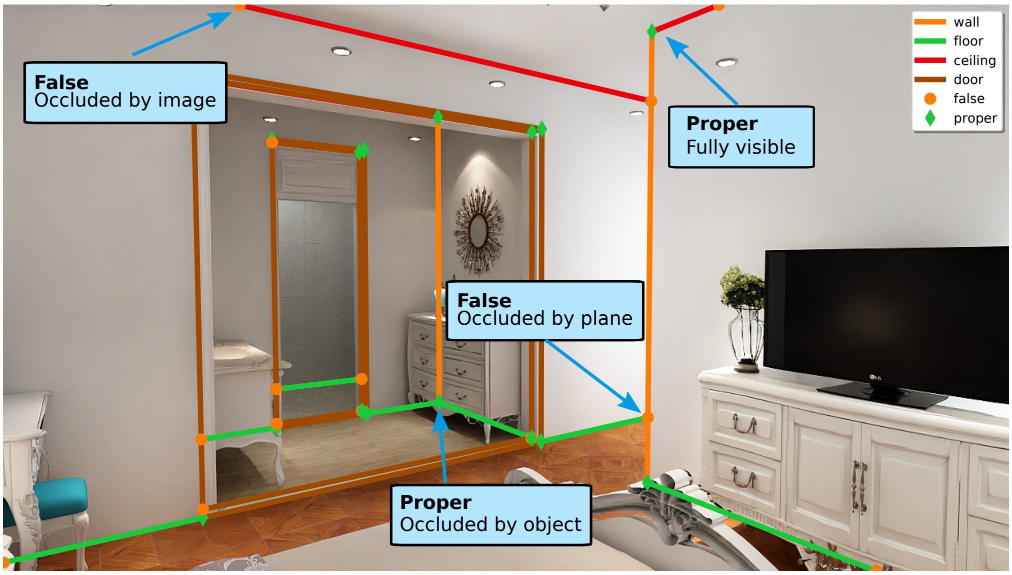


Figure 3: Paper II propose a wireframe representation for room layouts. The lines and points may be used for SfM and are stable since they depend on the room structure rather than appearance of furniture and wallpapers.

and flexible matching of points.

After the matching is done it is time to optimize all positions and camera poses to minimize the error in the map. This is called *bundle adjustment* and is an iterative process to reduce the *reprojection error*, which is the error obtained by projecting a key point or key object from the 3D model back to the photo and comparing the projection with the captured point or object. If using only keypoints, then the finished map is a set of points – also known as point cloud – which represents the surrounding structure.

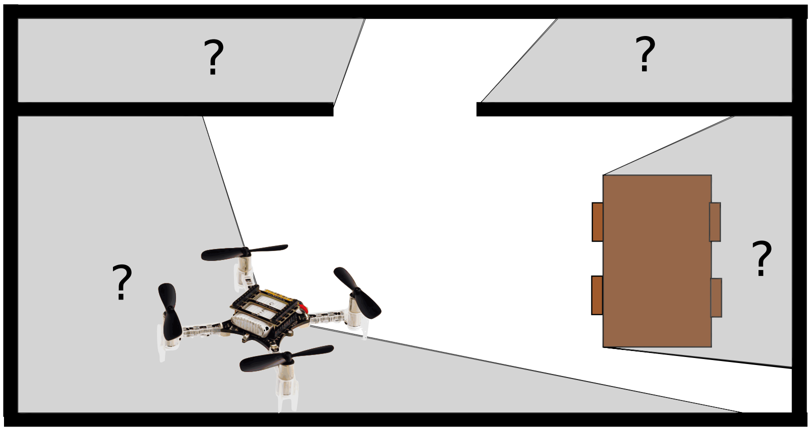
Performing bundle adjustment on a large-scale problem is difficult since the computations required do not scale proportionally to the number of images. Meaning that while we can process the outside of a building in a few minutes on a laptop it would take days to process a block with the same method. There are of course several ways to make it feasible, most often by splitting the problem into smaller pieces. This thesis studies *map merging*, which aims to as effectively as possible merge two or more point clouds without performing a new bundle adjustment. Methods are developed which optimize the reprojection error and do not require a full overlap of the points. It is useful for reducing computations but still flexible enough to correct shapes of objects if they are incorrect.

In short, this work touches on many aspects of the Structure from Motion pipeline. From detection of lines and polygons in images, to minimal solvers and merging of point clouds. Hopefully, the contributions may be a useful piece in the puzzle to enable autonomous systems and other services to understand our surroundings as we do.

Populärvetenskaplig sammanfattning

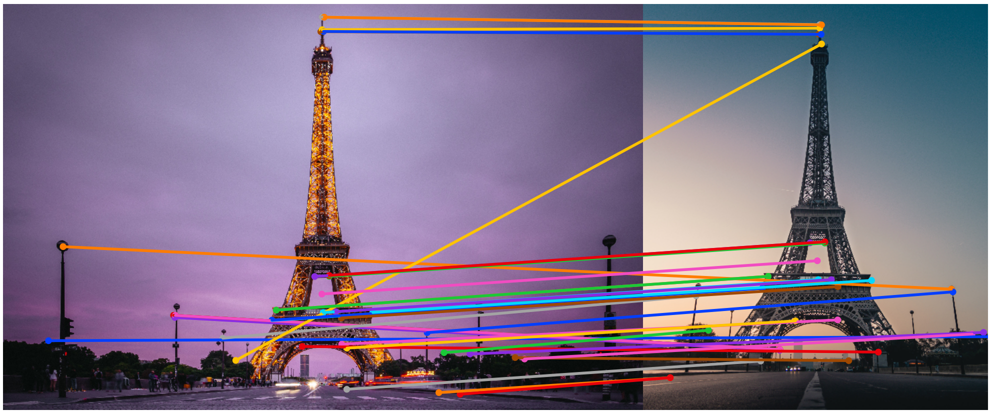
Som människor har vi förmågan att uppfatta alla tre spatiella dimensioner (3D) i vår omgivning, enbart med hjälp av våra ögon samt förflyttning av kroppen genom denna omgivning. Trots att våra ögon – liksom kameror – enbart ser en platt projektion av världen, så har vi lärt oss att tänka på den i 3D. Till exempel, genom att gå runt i ett hus lär vi oss hur rummen är placerade och dess storlek, genom att promenera runt ett kvarter kan vi få en känsla för var vi är, med hjälp av erfarenhet kan vi uppskatta höjden på ett skåp utan att ta fram måttbandet.

Kan vi lära en dator att göra detta? Detta arbete studerar närliggande frågor inom forskningsfältet *datorseende*. Som verktyg används geometriska matematiska modeller tillsammans med *neurala nätverk*, som är komplexa matematiska modeller inspirerade av hjärnans anatomi och behöver stora mängder data att lära sig ifrån. Exempel på frågor som studeras är: Kan vi träna ett neuralt nätverk att resonera kring dolda ytor inomhus som i Figur 1? Kan vi träna neurala nätverk att lista ut ett rums utformning i termer av golv, väggar och tak från en bild? Hur kan vi på ett effektivt sätt utnyttja rörelse för att skatta en modell i 3D från bilder tagna på olika platser?



Figur 1: En drönare i ett rum sett ovanifrån. Den försöker gissa vad som kan finnas dolt bakom väggarna och bordet.

Hjärnans förmåga till abstrakt tänkande är en av många faktorer som bidrar till vår orienteringsförmåga. För att orientera oss kan vi notera intressanta objekt – hus, skyltar, träd m.m. – som hjälper oss att förstå hur vi rör oss och ger oss möjlighet att hitta bättre när vi återbesöker samma plats. Inom datorseende finns det ett relaterat problem som kallas *Structure from Motion* (SfM), d.v.s. struktur från rörelse. Målet är att från bilder skapa en 3D-modell av omgivningen genom att beräkna kameras position och 3D-modellen samtidigt. Det första steget är typiskt sett att hitta intressanta objekt eller punkter i bilderna som kan användas för att relatera bilderna till varandra.

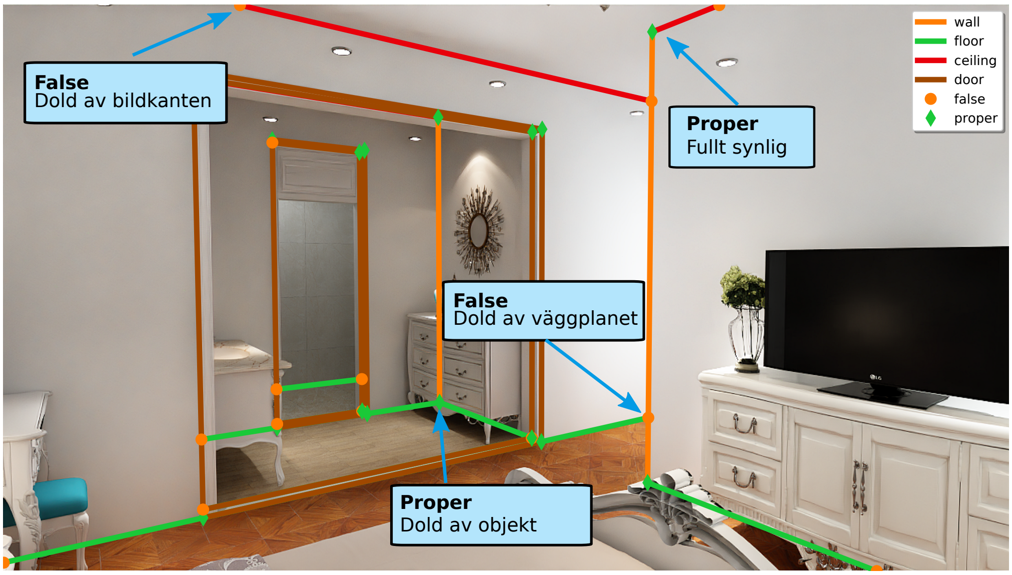


Figur 2: Två bilder tagna på Eiffeltornet vid olika tidpunkter och på olika platser. Intressepunkter (SIFT) har hittats i varje bild och en matchning baserad på utseendet har gjorts. Vi ser att vissa punkter matchar bra men delar av tornet saknar matchningar och ett flertal är helt felaktiga.
Bildkälla: Denys Nevozhai och Gautier Salles från `unsplash.com`.

Standardlösningen är att använda *intressepunkter* i bilden, med en unik färgsignatur som kallas *deskriptor*. Denna deskriptor används för att hitta matchande punkter i de andra bilderna. Detta är oftast en bra lösning men den har svårt att hantera upprepande mönster som tegelväggar eller blanka ytor som t.ex. vitmålade väggar. Förändringar över tid är också svårt, till exempel så ser ett träd på vintern väldigt annorlunda ut jämfört med ett träd på sommaren i Sverige. I Figur 2 ser vi automatiskt detekterade intressepunkter som är matchade mellan två bilder på Eiffeltornet, tagna vid olika tidpunkter. Under dessa förhållanden går det enbart att matcha vissa delar av tornet och många matchningar är felaktiga.

För att hantera dessa problem studeras i denna avhandling hur t.ex. träd och stolpar kan modelleras som parallella cylindrar för att förbättra robusthet och effektivitet. Dessutom studeras hur linjer och polygoner kan detekteras automatiskt och användas för att representera ett rums utformning i termer av golv, väggar, tak, fönster och dörrar som illustreras i Figur 3. Dessa objekt har en högre abstraktionsnivå än intressepunkter och kan förbättra robustheten i SfM, vilket är nödvändigt för att nå samma nivå av 3D-förståelse för en dator som för en människa.

När intressepunkter eller objekt har hittats är det dags att skatta position och orientering både av objekten samt av kamerorna som användes för att ta bilderna. För att hitta korrekta matchningar av objekt samt punkter mellan par av bilder behövs en robust matchningsmetod som kan ignorera de dåliga matchningarna. En vanlig metod för detta är *RANdom SAmple Consensus* (RANSAC), som löser ett *minimalt problem* för en mängd av slumpmässigt valda punkter, om och om igen. Det är viktigt att utveckla snabba *minimallösare* för dessa minimala problem, eftersom det möjliggör matchning av bilderna i realtid och i stor skala. Till exempel, för att skatta relativ position och orientering mellan två kalibrerade



Figur 3: Artikel II föreslår en wireframe-representation för rum. Linjerna och punkterna kan användas för SfM och är stabila eftersom de beror på rummets utformning snarare än dess utseende gällande t.ex. möbler och tapeter.

kameror så krävs minst fem matchande punkter i varje bild. Problemet har studerats av flertal forskare och det finns minimallösare som löser problemet på mikrosekunder. Detta innebär att vi kan köra tusentals iterationer i RANSAC för att hitta den bästa mängden punkter för matchning – på bara några millisekunder. I denna avhandling presenteras minimallösare för både matchning av parallella cylindrar samt flexibel matchning av intressepunkter.

När matchningen är gjord behöver position och orientering för alla punkter och kameror optimeras för att minimera felen i 3D-modellen, som vi också benämner *kartan*. Detta kallas för *bundle adjustment* och är en iterativ optimering som minskar *återprojiceringsfelet*, vilket är felet mellan den observerade intressepunkten och den skattade 3D-punkten projicerad tillbaka till bilden. Om enbart intressepunkter används så är den färdiga kartan en mängd av punkter i 3D – vilket kallas *punktmoln* – som representerar strukturen i omgivningen.

Att utföra bundle adjustment på ett storskaligt problem är svårt eftersom de nödvändiga beräkningarna inte skalar proportionerligt mot antalet bilder. Detta innebär att även om vi med en kan dator processa utsidan av en byggnad på några minuter skulle det ta dagar att modellera hela kvarteret med samma metod. Det finns förstås sätt att göra det möjligt, oftast genom att dela upp problemet i mindre bitar. I denna avhandling studeras *map merging*, där frågan är hur vi på ett effektivt sätt kan slå samman två eller flera punktmoln utan att göra nya bundle adjustments? Metoder presenteras som optimerar återprojiceringsfelet utan att kräva fullständigt överlapp av punktmolnen. De reducerar nödvändiga beräkningar men är fortfarande flexibla nog att korrigera formen på de ingående punkmolnen.

Sammanfattningsvis studeras flera aspekter av SfM i denna avhandling, från detektion av linjer och polygoner till minimallösare och sammanslagning av punktmoln. Förhoppningsvis kan dessa bidrag vara pusselbitar som bidrar till att autonoma system och andra tjänster kan förstå vår omgivning på samma sätt som vi gör.

List of Publications

This thesis is based on the following publications, referred to by their Roman Capital numerals. They are reproduced and included in this thesis with the permission of their respective publishers. The author's contributions to each paper is listed below. They are ordered according to the SfM pipeline, where Paper I-III operate directly on images, Paper IV solves for structure and camera poses, finally Paper V-VII covers the topic of map merging and takes SfM solutions as input.

I **In Depth Bayesian Semantic Scene Completion**
David Gillsjö, Kalle Åström
25th International Conference on Pattern Recognition (ICPR), 2021.

Author's contributions: DG suggested the project. DG developed the method and evaluation with help from KÅ. DG performed the experiments and implemented the necessary code, including model, training, and evaluation pipeline. DG wrote the paper with feedback from KÅ.

II **Semantic Room Wireframe Detection from a Single View**
David Gillsjö, Gabrielle Flood, Kalle Åström
26th International Conference on Pattern Recognition (ICPR), 2022.

Author's contributions: DG suggested the project. DG developed the method and evaluation with help from GF and KÅ. DG performed the experiments with help from GF. DG wrote the code, including model, training, and evaluation pipeline. DG wrote most of the paper, but all authors wrote some parts and gave feedback in the writing process.

III **Polygon Detection for Room Layout Estimation using Heterogenous Graphs and Wireframes**
David Gillsjö, Gabrielle Flood, Kalle Åström
Manuscript, Submitted to ICCV2023.

Author's contributions: DG suggested the project. DG developed the method and evaluation with help from GF and KÅ. DG developed and executed experiments as well as implemented the code for the model, training, and evaluation pipeline. DG wrote the paper with support from GF and KÅ.

IV **The Multi-View Geometry of Parallel Cylinders**
Erik Tegler, Johanna Engman, David Gillsjö, Gabrielle Flood, Viktor Larsson, Magnus Oskarsson, Karl Åström
Scandinavian Conference on Image Analysis (SCIA), 2023.

Author's contributions: All authors contributed equally to the project. The idea to develop minimal solvers for parallel cylinders grew out of discussions on earlier papers in the group. All authors were involved in the initial development of the theory. The ideas were refined in discussions with all authors and both theory and code gradually developed over time. All authors contributed with the writing of the paper.

V **Efficient Merging of Maps and Detection of Changes**
Gabrielle Flood, David Gillsjö, Anders Heyden, Kalle Åström
Scandinavian Conference on Image Analysis (SCIA), 2019.

Author's contributions: The idea for the merging algorithm came from KÅ, but was inspired from earlier papers in the group. All authors contributed to the development of the original idea to the theory and methods used in the paper. The code and the experiments were written and performed by GF, DG and KÅ. All authors contributed to the writing process.

VI **Generic Merging of Structure from Motion Maps with a Low Memory Footprint**
Gabrielle Flood, David Gillsjö, Patrik Persson, Anders Heyden, Kalle Åström
25th International Conference on Pattern Recognition (ICPR), 2021.

Author's contributions: The paper was a continuation and generalisation of Paper V. The ideas were developed over time and discussed between all authors. GF, DG, PP and KÅ wrote the code and did the experiments. DG performed the real data experiments together with PP. All authors contributed to the writing process.

VII **Minimal Solvers for Point Cloud Matching with Statistical Deformations**
Gabrielle Flood, Erik Tegler,David Gillsjö, Anders Heyden, Kalle Åström
26th International Conference on Pattern Recognition (ICPR), 2022.

Author's contributions: The paper was inspired from our earlier work on Paper VI and a Master's thesis by ET. The ideas were developed over time and discussed between all authors. All authors contributed with coding, experiments and writing of the paper.

Acknowledgements

With this thesis, a nearly seven-year long journey comes to an end. I have learned a great deal along the way and I am grateful to everyone who helped me make the journey. I'd like to thank my thesis supervisors *Kalle Åström*, *Anders Heyden* and *Gabrielle Flood* for the support and guidance over the years. Thank you *Kalle* for always supporting my ideas and helping me shape them into contributions for the research community; for sharing your ideas and introducing me to the algebra of projective geometry and minimal problems. Thank you for being an optimist and paving the road which allowed me to finally complete this thesis. *Anders*, thank you for your support and advice along the way as well as for proofreading of this thesis. Thank you *Gabrielle* for your help and advice with this thesis and for being a great colleague over the years. It has been a pleasure writing papers together with you. Thank you for staying positive and never shying away from drafts written in haste.

Many thanks to my other co-authors, *Patrik Persson*, *Erik Tegler*, *Johanna Engman*, *Viktor Larsson* and *Magnus Oskarsson*, for the collaboration over the years and for sharing your expertise and insights. A big thank you to all of my fellow *Ph.D. students* and *senior colleagues* at the Centre for Mathematical Sciences, especially to the members of the Computer Vision and Machine Learning group.

A warm thank you to my friends and relatives who have encouraged me to pursue this academic degree. I will name just a few of you, since this thesis is long enough as it is. Thank you *Christian Nelson*, for all the lunch and weekend discussions about family life and Ph.D. studies. My siblings *Josef* and *Elsa*, for being curious and supportive of my work and for the (occasionally) weekly gaming sessions between visits. My brother *Lukas*, for our technical discussions on polygons and for helping out in so many ways. We are glad you decided to move so close to us! My *parents*; thank you for promoting learning and teaching me how to study early on in life; for your support all these years and for the many road trips to Skåne – no matter how short the stay.

In addition to my thesis supervisors I'd like to thank *Nina and Ewen Macpherson*, *Lukas*, *Elsa* and *Jennifer* for your help with proofreading the thesis.

Along the journey to this thesis my two sons, *Hugo* and *Theodor*, came along for the ride. Even though this extended the journey, I'd like to thank you for contributing with your laughter and curiosity on the way. It has been a pleasure seeing you grow and exploring the world and I am looking forward to seeing more of if. Last but certainly not least I thank my lovely wife *Jennifer*, for encouraging me to start this journey and being a steadfast support along the way. Thanks for listening and picking me up when I hit a rough patch – there has truly been a few – and for celebrating with me when progress was made.

Funding

This thesis work was supported by the Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation. It is also supported by the strategic research project ELLIIT and the Swedish Foundation for Strategic Research project, Semantic Mapping and Visual Navigation for Smart Robots (grant no. RIT15-0038).

Contents

Chapter 1

Introduction

As humans, we are able to use our eyes and motion to explore and understand our surroundings. Even though we only perceive a projection of the world, we understand it has three spatial dimensions (3D). When we see an object, we may from experience extrapolate its extent and appearance in 3D. Or when we circle the object and see it from many views, we accumulate the information and form a 3D model in our mind.

As our technology advances we push the boundary with new hardware, making cameras, ranging, inertia and other sensors more accessible. It also enables powerful models capable of utilizing the growing processing power available. To make useful robots and services there is a growing need to utilize these sensors to make systems that may perceive the world around us and act according to our goals. Within the field of *Computer Vision* (CV) the research focuses on this perception problem; ”how can we make the computer see?”

The work presented in this thesis studies different parts of a 2D to 3D perception pipeline, taking multiple RGB images and reconstructing the surroundings in 3D by simultaneously localizing the cameras and the 3D structure. This is known as *Structure from Motion* (SfM) in the CV community or *Simultaneous Localization And Mapping* (SLAM) in the Robotics community. For example in Figure 1.1 pillars are positioned given two images using the methods developed in Paper IV, where silhouette lines of the pillars are detected and then used to solve for pillar positions and camera poses.

The first part of the SfM pipeline is to detect features that are related to the 3D structure and may be matched between viewpoints. Typically we work with point features, for example SIFT[36] *keypoints*, where each point is the center of a salient area in the image, for example a corner with a lot of gradient information. For each keypoint we extract descriptors that are feature vectors describing the area. We may then use these descriptors to match keypoints across images and find correspondences. Point features are not always the best choice, for

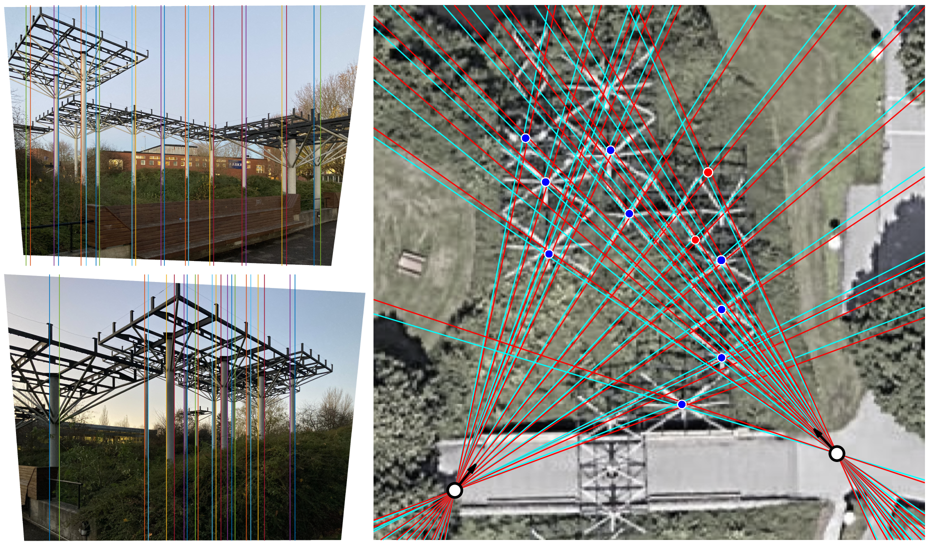


Figure 1.1: *Left:* two images with reprojected cylinder silhouettes, using the estimated solution. *Right:* the reconstruction registered to an top view. Also shown are the back-projected left and right silhouette lines in cyan and red. Note that there are no apparent image point correspondences in the two input images.

example if there is a lot of repeated structure (e.g. brick wall) or low texture environments (e.g. white wall) we may not get enough information. As seen in Figure 1.1, silhouette lines are sometimes a better choice. In Paper II and III we study how modern Deep Learning methods may be used to detect domain specific line and plane features, which may later be used to reconstruct an indoor room layout from multiple views. Figure 1.2 illustrates the wireframe representation we strive to detect automatically in Paper II.

The next part in the SfM pipeline is to solve for the transform between cameras and structure using robust estimation. It is a tricky matching and optimization problem since many features that look alike across images are not proper matches, known as outliers. A robust estimation method is able to ignore the outliers. A common choice is *Random Sample Consensus* (RANSAC), which solves for a large number of transforms on randomly sampled subsets of the detected geometric features. Here we are in need of minimal solvers, which are constrained to solve for a very specific transform, for example relative pose from 2 calibrated images and 5 point correspondences. The reason is that they are very fast and allows RANSAC to perform many iterations while keeping execution times down to allow for real time applications.

In Paper IV we use line detections in images (similar to Paper II) to reconstruct parallel cylinders – e.g. trees, poles and columns – from multiple views. We develop theory for related minimal problems and implement minimal solvers which can be used in a RANSAC setting. The solvers are evaluated in both simulated experiments and on real data.

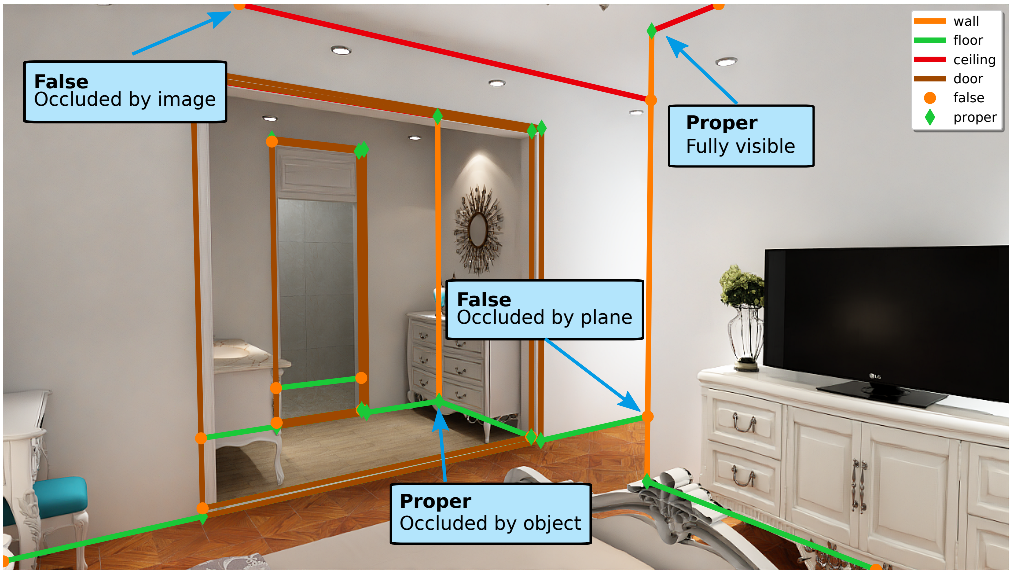


Figure 1.2: Paper II propose a wireframe representation for room layouts. The lines and points may be used for SfM and are stable since they depend on the room structure rather than appearance of furniture and wallpapers.

Given that we now have formed all 2D-to-2D and 2D-to-3D point feature matches necessary for a reconstruction, the geometry is optimized using a scheme known as *bundle adjustment*, which is an iterative approach to find the best solution in terms of re-projection error in the images. When bundle adjustment is completed, we have a 3D point cloud representing the 3D structure in our surroundings and 3D poses for all camera capture positions.

Since environments change we want to update our 3D structure, which we refer to as our map, to have good localization performance in other downstream applications. Then the question arises – if we now construct a new map, can we utilize the old information? – which is what we study in Paper V and VI. In these papers we develop theory and algorithms for map merging, the act of effectively merging these point clouds while minimizing the total re-projection error, without having to run bundle adjustment on all of the images from scratch. Due to drift, this may be difficult if the overlap between the point clouds is small. So in Paper VII we study how we may utilize statistical deformations to create new minimal solvers used for matching the point clouds.

With modern methods in Deep Learning we are able to utilize vast amounts of data to give our models flexible priors (experience) on what the world looks like. Given these priors, we may try to reconstruct a 3D representation directly from a single image. Paper I reconstructs a local 3D semantic voxel grid from a single RGBD image, which means a normal RGB image with added depth measurements. The task is to predict the semantic structure for both visible and occluded voxels, which may be of help for autonomous vehicles exploring an environment. The method is based on a *Bayesian Neural Network* (BNN), which is more transparent with uncertainties in the prediction compared to a deterministic Neural Network. To have a model with a reliable uncertainty estimate is essential for many applications, for example in autonomous vehicles.

The contributions of this thesis span both machine learning with Neural Networks and solving geometrical problems using algebraic geometry and optimization. Chapter 2 contains an introduction to concepts related to the Neural Network models in this work. Chapter 3 then goes into to mathematical modelling concepts of Structure from Motion and Map Merging. Finally, the scientific papers and their contributions are presented and discussed in Chapter 4.

Chapter 2

Learning with Images

Machine Learning and specifically *Deep Neural Networks* (DNN) has become an integral part of today's Computer Vision pipelines due to their superior performance in detection and classification compared to other methods. While Neural Networks has been studied since the 1940s [18], the real breakthrough in Computer Vision tasks came in 2012 with the *Convolutional Neural Network* (CNN) named AlexNet [31]. It was trained on ImageNet [12] for the image classification task and outperformed previous methods by a large margin. CNNs and other DNN architectures has since then been the dominating models in the Computer Vision research community.

Paper I-III in this thesis develop CNN and Graph Neural Network architectures for various tasks and this chapter contains a brief overview on central concepts when it comes to these models.

2.1 Neural Network Structure

Deep Neural Networks are built by passing the input signal through several layers that extract features and then use these for the task at hand, often regression or classification. The strength of the DNN is that it is a learned non-linear model that may approximate a broad range of functions according to the *Universal Approximation Theorem* [25] and its many variations. One type of DNN is the Convolutional Neural Network model, which is illustrated in Figure 2.3. Features are extracted with hidden CNN layers and followed by hidden *Multilayer Perceptron* (MLP) layers that give a final score vector used to classify the image. The model is trained for the MNIST dataset which contains digits 0-9, hence the output vector holds ten different scores, one for each digit. The following sections will introduce layers commonly used in Deep Neural Networks and in the papers.

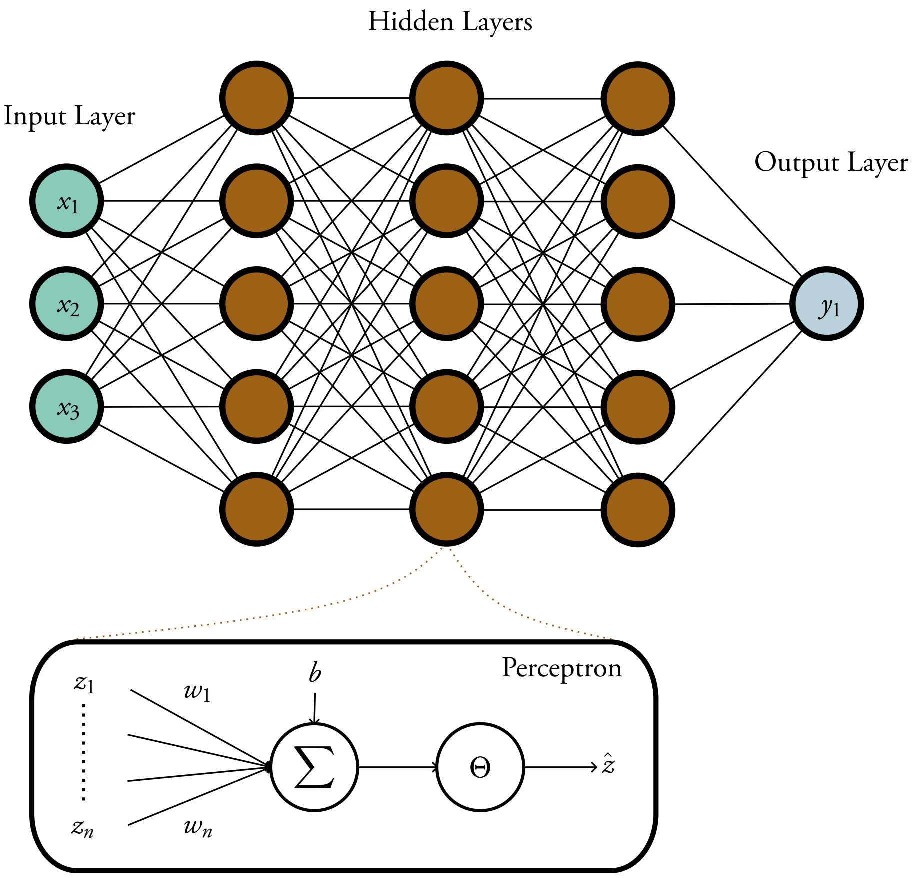


Figure 2.1: An illustration of an MLP with fully connected perceptrons. The bottom box illustrates how each perceptron forms a weighted sum passed through a non-linear activation function Θ.

2.1.1 Multilayer Perceptron

The Multilayer Perceptron (MLP) is built by combining linear *Fully Connected* (FC) layers with a non-linear *Activation Function* [18], as illustrated in Figure 2.1. The first layer is called the input layer, then follows a number of hidden layers with perceptrons and finally an output layer. A layer i may be described as a function

$$\mathbf{x}_{i+1} = f_i(\mathbf{x}_i, \mathbf{W}_i, \mathbf{b}_i) = \mathbf{W}_i\mathbf{x}_i + \mathbf{b}_i, \tag{2.1}$$

where $\mathbf{x}_i$ is the input vector of length N_i, $\mathbf{W}_i$ are weights with dimension $(M_i \times N_i)$ and $\mathbf{b}_i$ bias with length M_i. Here, N_1 is given by the input data $\mathbf{x}_1$ to the first layer, for the subsequent layers we have $N_{i+1} = M_i$. The non-linearity enters as an activation function Θ between FC layers. The most common choice is to use *Rectified Linear Unit* (ReLU), but there are many other options. Now we may build an MLP $\mathcal{G}$ with D layers, also referred

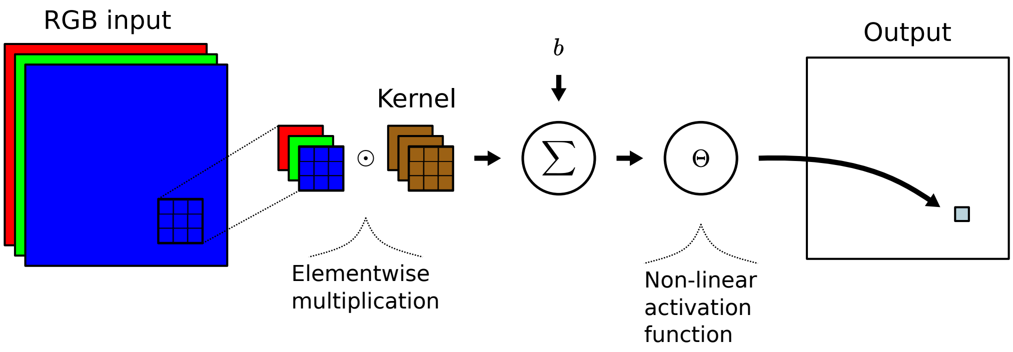


Figure 2.2: Illustrates the principle of a CNN layer with an RGB input image and one output channel. For each output pixel, a 3×3 kernel is applied to corresponding input pixel and the cross correlation is calculated. Finally the result is passed to a non-linear activation function and the result is stored in each kernel's corresponding output layer.

to as depth, by stacking the functions

$$\mathcal{G}(\mathbf{x}) = \Theta(f_D(\Theta(f_{D-1}(....\Theta(f_1(\mathbf{x}))))), \tag{2.2}$$

where M_i may be unique for each layer. Hence the depth D and layer dimension M_i governs the complexity of the model. If the goal is to perform regression we set M_D equal to the dimension of the target, while if we are solving a classification problem we set M_D to the number of classes (including background) and apply the softmax operator

$$\text{softmax}(\mathbf{x})_k = \frac{e^{x_k}}{\sum_j e^{x_j}}, \tag{2.3}$$

where $k = 1, ..., M_D$. The output is a score where M_D elements are real and sum to one, which means that they can be interpreted as a probability density function.

2.1.2 Convolutional Neural Network

For a deep CNN the principle of layers is the same as for the MLP. A CNN typically takes input of 1,2 or 3 dimensions which for example could be signal over time, image or 3D voxel grid respectively. For illustration we focus on the 2D case using RGB image of dimension $H \times W$ as input as shown in Figure 2.2. Now the input $\mathbf{x}_1$ has dimension $3 \times H \times W$, where the first dimension corresponds to the number of colour channels. Each CNN layer i has $j = 1, 2, ..., J_i$ kernels $\mathbf{K}_{i,j}$ with dimension $N_i \times M_i \times M_i$. These are applied to the input using cross correlation (convolution without flipping the kernel). Cross correlation and convolution is essentially the same – since the network can learn the flipped kernel – but cross correlation is chosen because it is simpler to formulate. The convolution layer i is computed as

$$\mathbf{x}_{i+1} = h_i(\mathbf{x}_i) = \mathbf{b}_i + \left[\sum_{c=1}^{N_i} \mathbf{K}_{i,1}(c) \star \mathbf{x}_i(c) \quad \dots \quad \sum_{c=1}^{N_i} \mathbf{K}_{i,J_i}(c) \star \mathbf{x}_i(c) \right], \tag{2.4}$$

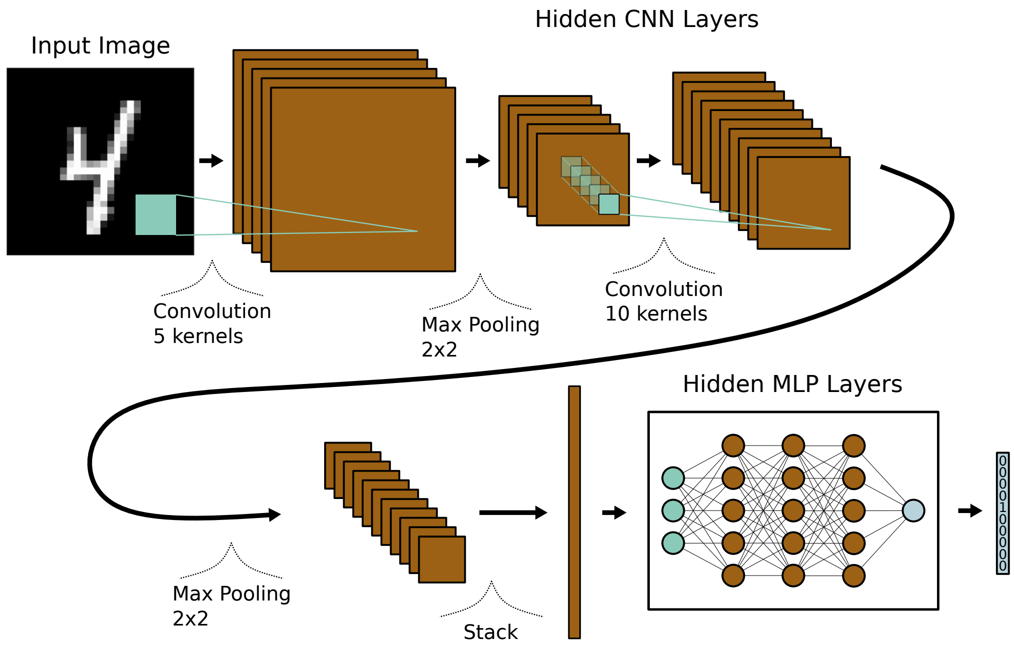


Figure 2.3: Illustrates the principle of a CNN architecture for image classification. An image of a digit is first put through some hidden CNN layers alternating convolution and max pooling. Prior to the MLP layers the CNN output is flattened to a vector and the MLP output is a vector with 10 elements indicating the likelihood of each digit.

where $\mathbf{x}_i(c)$ denotes channel c of $\mathbf{x}_i$ and $\star$ denotes the cross correlation operator. The bias $\mathbf{b}_i$ has dimension $J_i = N_{i+1}$, i.e. one scalar per kernel and therefore output channel. To simplify notation the addition indicates that the bias scalar connected to each kernel is added to each element of the respective output channel. These layers may now be stacked using activation functions and pooling layers, see Section 2.1.5 and 2.1.6, respectively. This forms a network architecture, for example as illustrated in Figure 2.3, where the output channels for layer i is identical to the number of kernels J_i. If only using activation functions we may form a network $\mathcal{H}(\mathbf{x})$ of depth D as in the MLP case,

$$\mathcal{H}(\mathbf{x}) = \Theta(h_D(\Theta(h_{D-1}(....\Theta(h_1(\mathbf{x})))), \tag{2.5}$$

by defining the number of kernels and their size for each layer.

2.1.3 Graph Convolutional Network

A *Graph Convolutional Network* (GCN) is a generalization of the CNN that may be applied to graphs. GCNs may be used for different tasks, the most common being graph classification, node classification or link prediction. For node classification each node is assigned a separate label at inference, as illustrated in Figure 2.5, while in graph classification the

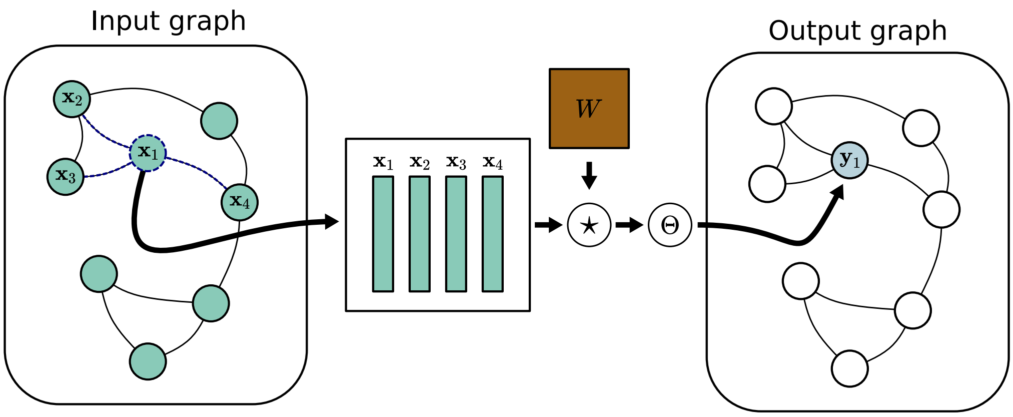


Figure 2.4: Shows the principle of a Graph Convolutional Network layer. The output feature vector $\mathbf{y}_1$ is created by an approximate graph convolutional operation between a weight matrix W and the input features $\mathbf{x}_1, \ldots, \mathbf{x}_4$, which are the nodes own input feature and the neighbours input features.

whole structure is assigned a label. The goal for link prediction is to predict the likelihood of connection between nodes and is for example used in recommendation algorithms. In this thesis, node classification is used in Paper II-III and link prediction in Paper III.

While there are many types of GCN layers [55], this section will focus on the GCN layer used in Paper II, which is developed by Kipf and Welling [30]. Graph convolutions can be divided into spatial and spectral methods and this model is the latter. The GCN model makes a first order approximation of spectral convolution, resulting in each outgoing node feature being a learned combination of the input node feature and its 1-hop neighbours as illustrated in Figure 2.4.

Let $X \in \mathbb{R}^{N \times M}$ denote the input features for N nodes with feature vector size M. The proposed GCN layer forms an output Y from input X as

$$Y = \Theta\left(\bar{D}^{-\frac{1}{2}}\bar{A}\bar{D}^{-\frac{1}{2}}XW\right), \tag{2.6}$$

where W is a learned weight matrix. The extended adjacency matrix $\bar{A} = A + I$ is the adjacency matrix of the input graph with added self-connections by addition of the identity matrix I. Finally we have $\bar{D} = \sum_j \bar{A}_{i,j}$ and Θ is a non-linear activation function, analogous to the CNN and MLP layer.

The reasoning for this model comes from spectral graph convolutions, where a signal $x \in \mathbb{R}^N$ for N nodes may be convolved with a function $g_\theta = \mathrm{diag}(\theta)$ parametrised by $\theta \in \mathbb{R}^N$ in the Fourier domain as

$$g_\theta \star x = U g_\theta U^T x. \tag{2.7}$$

Here U is a matrix of eigenvectors of the normalized graph Laplacian $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}} = U\Lambda U^T$, where $U^T x$ is the graph Fourier transform of x, $D = \sum_j A_{i,j}$ and Λ a diagonal

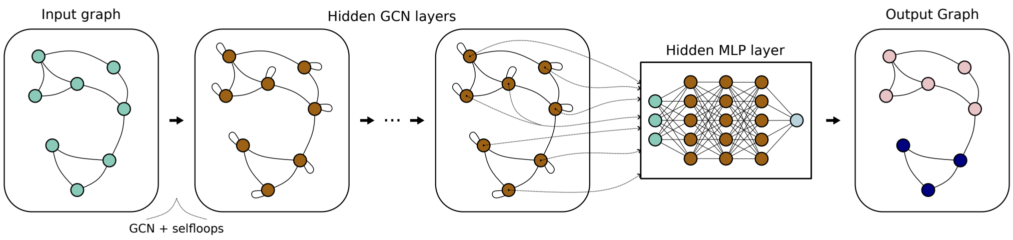


Figure 2.5: Illustrates the principle of a GCN architecture for node classification. An input graph with node features is extended with self loops and then passed through a series of GCN layers. Each node feature vector in the resulting graph is then fed to an MLP classifier to predict the node label.

matrix of eigenvalues. The graph convolution in Equation (2.7) is expensive to compute and may be approximated using Chebyshew polynomials $T_k(x)$ up to the Kth order as

$$g_\theta \star x \approx \sum_{k=0}^{K} \hat{\theta}_k T_k \left(\frac{2}{\lambda_{\max}} L - I \right) x, \tag{2.8}$$

where $\hat{\theta}$ are the Chebyshew coefficients and $\lambda_{\max}$ the largest eigenvalue of L. The order K here governs the size of the neighbourhood utilized during the convolution. By choosing $K = 1$ and assuming $\lambda_{\max} = 2$, since the Neural Network can handle the scaling, the approximated convolution is

$$g_\theta \star x \approx \hat{\theta}_0 x - \hat{\theta}_1 D^{-\frac{1}{2}} A D^{-\frac{1}{2}} x. \tag{2.9}$$

The final approximation is reached by reducing the complexity further, setting $\theta = \hat{\theta}_0 = -\hat{\theta}_1$ and utilizing the re-normalization trick, resulting in

$$g_\theta \star x \approx \theta \bar{D}^{-\frac{1}{2}} \bar{A} \bar{D}^{-\frac{1}{2}} x. \tag{2.10}$$

The final expression in Equation (2.6) is gained by extending to a multi-channel input X.

2.1.4 Heterogeneous Graph Networks

Compared to the homogenous graphs used in the previous section, the *heterogeneous graph* (HG) allows for different types of nodes and edges. For example in Figure 2.6 a polygon is represented as a HG with node types end point, line segment and polygon centroid. To construct a Neural Network layer the method must account for how the different node types should be weighted against each other in the neighbourhood. In Paper III we utilize the *Heterogeneous Graph Transformer* (HGT) [26], which is a type of Heterogeneous Graph Network model using attention based message passing. Let t be the target node for which we need to calculate a new feature $Y[t]$, then

$$Y[t] = \underset{\forall s \in N[t]}{\text{Aggregate}} \left(\text{Message}(s) \cdot \text{Attention}(s, t) \right). \tag{2.11}$$

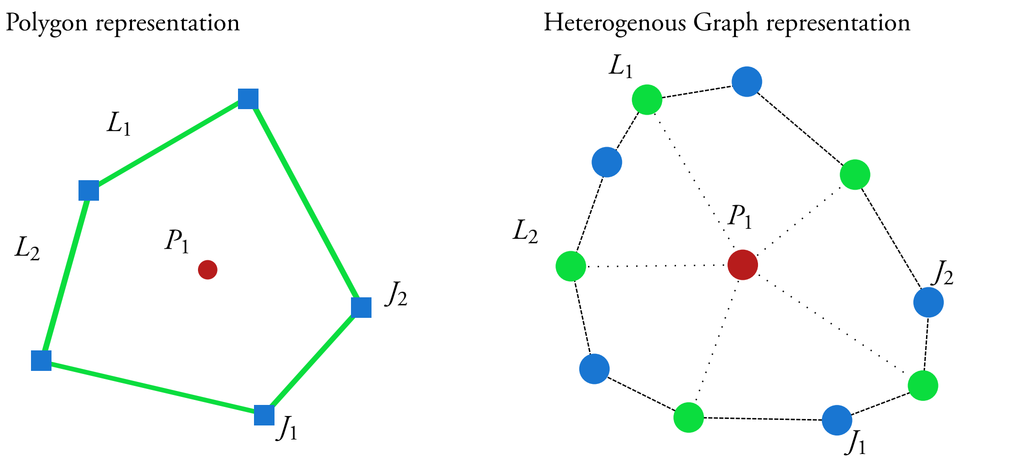


Figure 2.6: Illustration of how a polygon (left) may be represented with a heterogeneous graph (right) consisting of node types end point, line and polygon centroid. In this illustration there are two types of edges, junction-line and line-plane.

Here, $N[t]$ is the neighbourhood of t, which in most cases is the 1-hop neighbourhood, just like for GCN in the previous section. Message(s) might just be the incoming feature $X[s]$ for node s, but for the HGT it is the result of a linear operation on incoming feature $X[s]$, which parameters depend on the node type $\tau(t)$ and edge type $\theta(e)$. The Aggregate operation may be a sum, mean or other operation that suits the application, for the HGT it depends on the target node t. The HGT model calculate different attention weights Attention(s, t) based on the meta relation $\langle\tau(t), \theta(e), \tau(s)\rangle$. For more details on the model, including aggregation and message construction, see the paper by Hu *et al.*[26].

2.1.5 Activation Function

An essential part of the neural network models ability to approximate complex functions is the non-linear responses to the input. To achieve this, each layer of the network typically ends with a non-linear activation function. The most common for hidden layers is the *Rectified Linear Unit* (ReLU) $y = \max(0, x)$. The downside to this function is that the gradient may easily go to zero during back-propagation and push neuron weights to zero, making them inactive. An alternative is to use the *Softplus* function $y = \log(1 + e^x)$, which is a smooth approximation of ReLU and does not go to zero as quickly. Both of these functions are shown in see Figure 2.7 for comparison.

When is comes to the output layer we want choose different functions depending on the task. For regression or binary classification tasks the most common choice is the *Sigmoid* function $y = \frac{1}{1+e^{-x}}$, which maps the output to the range $[0, 1]$. For multi-label classification we need a score for each class and therefore use the *Softmax* function $y_i = \frac{e^{x_i}}{\sum_j e^{x_j}}$,

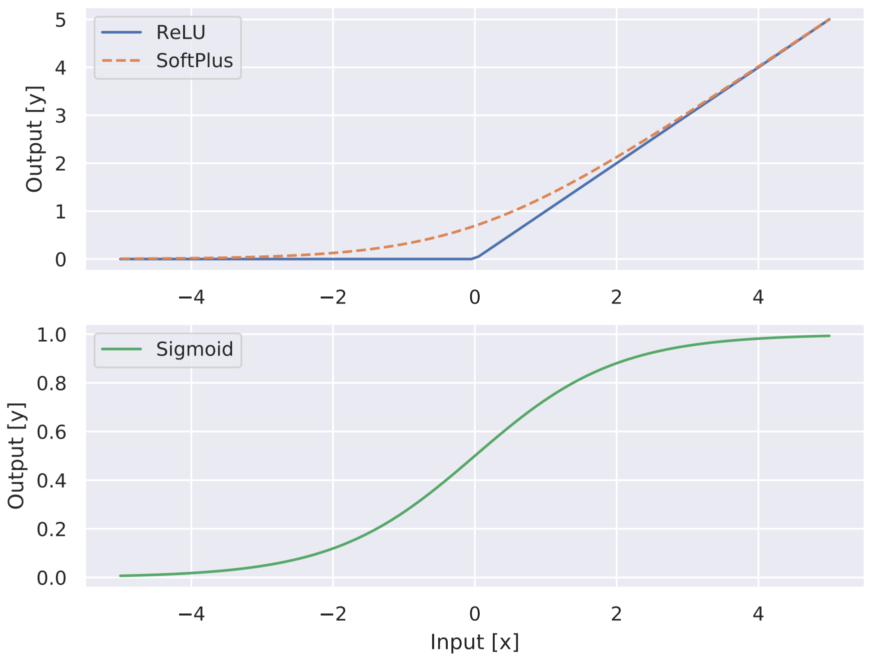


Figure 2.7: Here we see the most common activation functions. The top shows ReLU and Softplus for inner layer activation. The bottom shows Sigmoid for output activation.

which gives each class i a positive score y_i such that $\sum_i y_i = 1$.

To see that both may be used for binary classification – e.g. background vs foreground – consider the following. If the network yields a single output logit x for sigmoid or the logit vector $\begin{bmatrix}0 & x\end{bmatrix}$ for softmax we get

$$\mathrm{Sigmoid}(x) = y_s = \frac{1}{1+e^{-x}}, \tag{2.12}$$

$$\mathrm{Softmax}(\begin{bmatrix}0 & x\end{bmatrix}) = \begin{bmatrix}\frac{1}{1+e^x} & \frac{e^x}{1+e^x}\end{bmatrix} = \begin{bmatrix}\frac{e^{-x}}{1+e^{-x}} & \frac{1}{1+e^{-x}}\end{bmatrix} = \begin{bmatrix}1-y_s & y_s\end{bmatrix}. \tag{2.13}$$

Hence, if we interpret the sigmoid output y_s as a probability for the foreground class, then the probability of the background class must be $1 - y_s$, which is the same probabilities as the softmax operator yields.

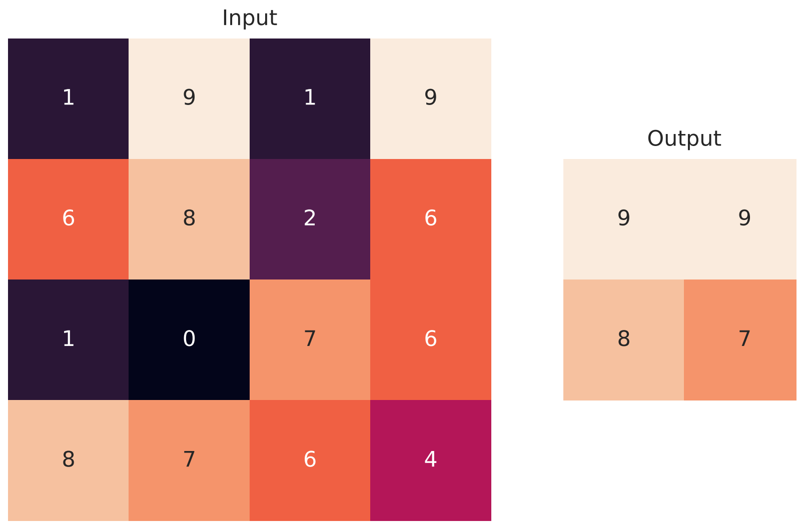
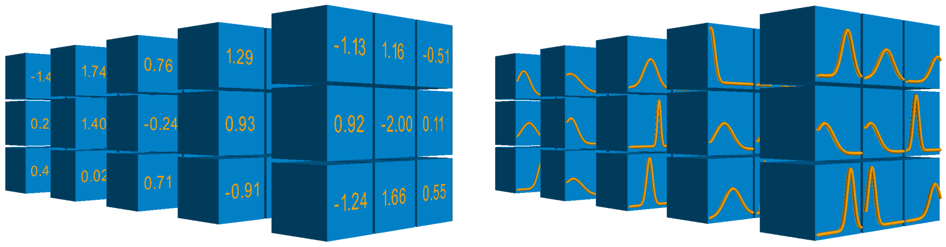


Figure 2.8: Illustration of the max pooling operation. To the left is the input and to the right we have the output of a max pooling operation with kernel size 2 and stride 2. The upper left pixel in the output correspond the the upper left 2x2 patch in the input.

2.1.6 Pooling

To allow the Neural Network to reason about the input on a higher level, i.e. over several time steps or larger spatial area, *pooling layers* are frequently used. The pooling operation exists for both CNNs and GCNs but is more straight forward in the CNN case and the papers in the thesis do not use GCN pooling. The most common types for CNNs are Max-, Average- and Min-pooling and they are applied after the activation layer. For the remainder of this section Max-pooling will be explained, but the concept is easily applied to the other types. Much like the convolution layers we have a kernel, but this time the kernel takes the maximum value per channel over all affected pixels. For pooling we introduce the *stride* parameter, which is the step size taken when computing the next output pixel. Hence, the if stride is larger than one, the output will have smaller dimension than the input. For pooling the stride is typically the same as the length of the side of the kernel. Stride can also be applied in a convolution layer. In Figure 2.8 we see an example of a 2x2 kernel which is applied with stride 2 on a 4x4 input. The output will be the maximum value for each of the quadrants where we apply the kernel. The output size W_{out} of a square image with input width W_{in} is

$$W_{\text{out}} = \left\lfloor \frac{W_{\text{in}} - \text{kernel\_size}}{\text{stride}} + 1 \right\rfloor . \tag{2.14}$$



(a) Standard (b) Bayesian

Figure 2.9: In 2.9a we see a filter bank from a standard 2D CNN, each weight is a scalar. In 2.9b we see a filter bank in a Bayesian Variational Inference 2D CNN, here each weight represented as a distribution which is sampled from at inference time.

2.2 Bayesian Networks

In Paper I we extend a CNN for Scene Completion to a Bayesian CNN, which is a CNN model which predicts a distribution instead of a single output at inference time. The main reason for doing this is that Neural Networks in their standard setting are prone to over-fitting and easily become over-confident in their predictions, even though the input may be well outside the domain of the training data. Using Bayesian methods to model the network weights and regularization is one way of approaching these issues. In the paper we use *Bayes by Backprop* [5, 48], which is based on *Variational Inference*. This section briefly introduces the concept.

Consider a neural network as a probabilistic model $P(\mathbf{y}|\mathbf{x}, \mathbf{w})$, where $\mathbf{x}$ is the input, $\mathbf{y}$ output and $\mathbf{w}$ are network weights. To learn $\mathbf{w}$ from a set of N training examples $\mathcal{D} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), ..., (\mathbf{x}_N, \mathbf{y}_N)\}$ without regularization is equivalent to the *Maximum Likelihood Estimator* (MLE) given by

$$\mathbf{w}^{MLE} = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w}). \tag{2.15}$$

We introduce regularization by placing a prior on the weights and take the *Maximum a Posteriori* (MAP) estimate

$$\mathbf{w}^{MAP} = \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) = \arg\max_{\mathbf{w}} \left(\log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w})\right), \tag{2.16}$$

where we achieve L_2 regularization, or weight decay, by choosing a Gaussian prior $P(\mathbf{w})$.

The BNN extends the probabilistic model by estimating the *posterior distribution* over the weights $P(\mathbf{w}|\mathcal{D})$. Let $P(\mathbf{w}|\mathcal{D})$ be the posterior for our Bayesian Neural Network. Given the posterior we can make predictions for the output $\hat{\mathbf{y}}$ given unseen data $\hat{\mathbf{x}}$ by taking the expectation over the posterior

$$P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) = \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}\left[P(\hat{\mathbf{y}}|\hat{\mathbf{x}},\mathbf{w})\right] = \int P(\hat{\mathbf{y}}|\hat{\mathbf{x}},\mathbf{w})P(\mathbf{w}|\mathcal{D})\,d\mathbf{w}. \tag{2.17}$$

This is not tractable since we would need to integrate over all possible weights in the network. The idea is to approximate the posterior distribution over weights by modelling them as independent Gaussian random variables. We then use Variational Inference tools to optimize our neural network to fit the proper distribution as well as possible. In Figure 2.9 we see an illustration of a standard CNN filter bank vs a Bayesian CNN filter bank. Since the weights are sampled during inference we get a slightly different network for each input. This enables robust estimations by averaging, much like an ensemble[1], while training a single model. The downside is that we have doubled the amount of parameters since each weight is characterized by a mean and standard deviation.

We estimate the posterior using a simpler model $q(\mathbf{w}|\theta)$ with learnable parameters θ, which minimizes the Kullback-Leibler (KL) divergence to the true posterior, i.e.

$$\begin{aligned}
\theta^* &= \arg\min_\theta KL[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] && (2.18)\\
&= \arg\min_\theta \int q(\mathbf{w}|\theta)\log\frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} && (2.19)\\
&= \arg\min_\theta KL[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})]. && (2.20)
\end{aligned}$$

This cost function is known as variational free energy or expected lower bound. We denote it as

$$\mathcal{F}(\mathcal{D},\theta) = KL[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})], \tag{2.21}$$

where the first part is the complexity cost as it enforces model simplicity using the prior. The second part is the likelihood, which describes how well the model describes the data.

Using Monte Carlo sampling the cost in Equation (2.21) is approximated as

$$\mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n}\frac{\beta}{n}\left[\log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)})\right] - \log P(\mathcal{D}|\mathbf{w}^{(i)}), \tag{2.22}$$

where $\mathbf{w}^{(i)}$ is a sample from the variational posterior $q(\mathbf{w}^{(i)}|\theta)$. This approximation enables more priors and posteriors since a closed form solution for the first term in Equation (2.21) is not necessary. The scale factor $\frac{\beta}{n}$ with β as design parameter is introduced to tune the amount of regularization from the complexity cost.

[1]a set of NNs trained from different initialization weights

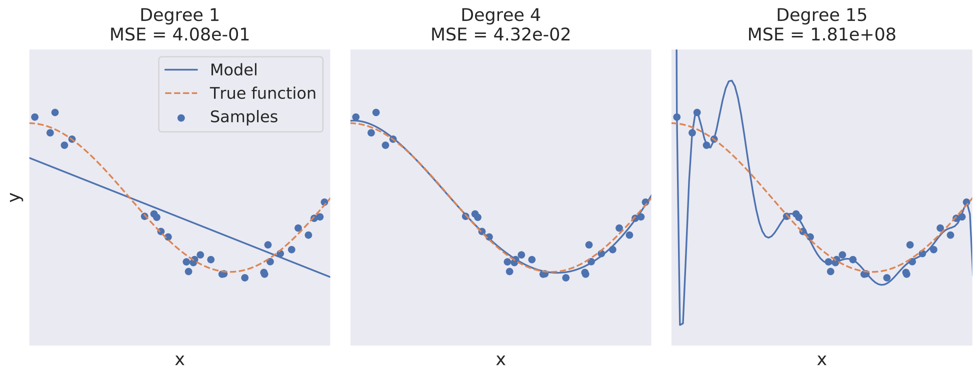


Figure 2.10: Example of overfitting and underfitting a model. To the left the model is only of degree 1 and not flexible enough, therefore underfitting the data. In the middle the degree is increased to 4, which is sufficient to approximate the underlying function. To the right the degree is increased to 15, causing the model to overfit. The MSE is calculated on a validation dataset (not shown here) and indicates generalization performance.

2.3 Training the Model

There are numerous ways to learn a model from data, but in this thesis we use *supervised learning*. Which means that all data is annotated with the desired output of the model. After defining the model it is time to learn from the annotated data acquired for the task at hand. This is done by defining *loss functions* that will guide the model to perform the task. The loss is minimized through an iterative optimization process where we loop through the data. The following subsections touches on the different aspects of training a deep neural network, for more details see Goodfellow *et al.*[18].

2.3.1 Split the Data

One major challenge in parameter estimation from data is overfitting. This is especially true when it comes to complex models such as Deep Neural Networks, which are very flexible by design. In Figure 2.10 we see an example, where a polynomial model is fitted to samples from a cosine function. To the left the model is only of degree 1 and not flexible enough, therefore underfitting the data. In the middle the degree is increased to 4, which is sufficient to approximate the underlying function. To the right the degree is increased to 15, causing the model to overfit to the data not generalizing well, meaning that the performance between the sample points, i.e. training data, is poor.

To detect this during training it is necessary to evaluate the performance on validation data, which is a set of the data that is withheld from training. By calculating a loss or performance metric on this data, for example the MSE in Figure 2.10, we notice when the model starts to overfit. Other benefits of having a validation set is that we may use that for tuning of hyperparameters that affect the training.

It is also customary to have a second hold out set called a test set. The purpose of this dataset is to fairly compare methods. This allows researchers to tune hyper-parameters on the validation set and then make an unbiased comparison on the test set. The split may vary, but the training set is the largest. For example COCO [34] has a $(\frac{1}{2}, \frac{1}{4}, \frac{1}{4})$ split while Structured3D [56] has $(\frac{10}{12}, \frac{1}{12}, \frac{1}{12})$.

While overfitting is an issue for most implementations, it has been shown [43] that DNN models sometimes suddenly increase in validation set performance long after they stagnate in training set performance. This behaviour is termed *grokking* and shows that there is still a lot we do not understand about the training of these complex and over-parameterized DNN models.

2.3.2 Optimization

When optimizing a DNN model, it is typically intractable to optimize directly toward the target metric, for example *Average Precision* (AP) [34]. Therefore we define a cost function $J(\mathbf{w})$ over the network weights $\mathbf{w}$ to minimize, which we hope will also optimize the target metric. We do this by minimizing the expected loss on the training set

$$J^*(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} L(f(\mathbf{x}_i, \mathbf{w}), \mathbf{y}_i), \tag{2.23}$$

where L is a loss function, f is the DNN model, $\mathbf{x}_i$ is a data sample from our training data and $\mathbf{y}_i$ is the respective annotated target.

Stochastic Gradient Descent

A straightforward optimization method is to use a gradient descent method to iteratively find a local minimum. Since it is a sum we may easily compute it as

$$\nabla_{\mathbf{w}} J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N} \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, \mathbf{w}), \mathbf{y}_i). \tag{2.24}$$

However, calculating this gradient for all training samples for each optimization step is not feasible since these models require a lot of data. For example ImageNet contains more than 14 million images. So to optimize these models we use *Stochastic Gradient Descent* (SGD).

For SGD we choose a sample size $M << N$, then take a random subset of size M – also called a *minibatch* – at each gradient update step and perform the parameter update

$$\hat{\mathbf{g}} \leftarrow \frac{1}{M} \sum_{i=1}^{M} \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, \mathbf{w}), \mathbf{y}_i), \tag{2.25}$$

$$\mathbf{w} \leftarrow \mathbf{w} - \epsilon_l \hat{\mathbf{g}}, \tag{2.26}$$

where ϵ_l is the *learning rate*. Each update by the optimizer it called a *batch* (short for minibatch) and the training data is typically sampled without replacement until all samples are depleted. Each pass through the training data is called an *epoch* and for most applications it is not enough to loop through the data once. Instead we loop through the data until we no longer see any gains on the validation data. To give an example of the scale, in Paper II we train for 40 epochs and each epoch contains around 150'000 images.

The learning rate is tuned specifically for the model and the problem, often by trial and error. The trade-off here is that a high learning rate makes for a faster training and also adds some regularization to the solution. On the other hand it may be unstable and start to oscillate due to the uncertainty in the gradient estimate, therefore overshooting the minimum. Hence, a smaller learning rate is often required to get the best performance. There are several schemes for how to adjust the learning rate during training, the simpler ones being either (*i*) linear decrease each epoch towards a target learning rate or (*ii*) reduction by a factor at a fixed interval of epochs.

Momentum

A common problem with these large NN models is that the batch size M for SGD is often small, since the model and the sample data needs to fit in GPU memory. This causes the gradient estimate to be very noisy. A way to alleviate this problem is to use a low pass filter to keep track of the general direction of the gradient. This is known as a *momentum* update [45]. The idea is to track a general velocity of the weights, denoted $\mathbf{v}$, which is updated with the gradient just as in standard SGD. The new update rule is then

$$\hat{\mathbf{g}} \leftarrow \frac{1}{M} \sum_{i=1}^{M} \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, \mathbf{w}), \mathbf{y}_i), \tag{2.27}$$

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \epsilon_l \hat{\mathbf{g}}, \tag{2.28}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{v}, \tag{2.29}$$

where α controls how dynamic the filter is. Choosing $\alpha = 0$ yields the SGD again, while a large α will adapt very slowly to changes in the gradient. Common values are 0.5, 0.9 and 0.99.

Adam

There are several methods to adaptively adjust the learning rate during optimization. Adam [29] is a popular method which is fairly robust when it comes to the choice of parameters. Adam utilizes both the first and second moment of the gradient and adds a bias correction term. The update looks as follows

$$\hat{\mathbf{w}} \leftarrow \mathbf{w} + \mathbf{v}, \tag{2.30}$$

$$\hat{\mathbf{g}} \leftarrow \frac{1}{M} \sum_{i=1}^{M} \nabla_{\mathbf{w}} L(f(\mathbf{x}_i, \hat{\mathbf{w}}), \mathbf{y}_i), \tag{2.31}$$

$$\mathbf{r} \leftarrow \rho \mathbf{r} (1 - \rho) \hat{\mathbf{g}} \odot \hat{\mathbf{g}}, \tag{2.32}$$

$$\mathbf{v} \leftarrow \alpha \mathbf{v} - \frac{\epsilon_l}{\mathbf{r}} \odot \hat{\mathbf{g}}, \tag{2.33}$$

$$\mathbf{w} \leftarrow \mathbf{w} + \mathbf{v}, \tag{2.34}$$

where $\odot$ is elementwise multiplication, α and ρ governs the update rate for the first and second moment respectively.

2.3.3 Loss Functions

While there are many tasks for which we may adapt a NN, in this thesis the applications are either classification or regression. This section gives a brief introduction of the concepts and how to construct the loss functions.

Regression

In regression the task is to predict a numerical value on $\mathbb{R}$, so we want to find a function $f \colon \mathbb{R}^n \to \mathbb{R}$. As in Section 2.2 we may use Maximum Likelihood reasoning to find a useful loss function.

Consider a neural network as a probabilistic model $P(y|\mathbf{x}, \mathbf{w})$, where $\mathbf{x}$ is the input, y output and $\mathbf{w}$ are network weights. To learn from a set of N training examples $\mathcal{D} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), ..., (\mathbf{x}_N, y_N)\}$ we use the Maximum Likelihood estimator for the optimal network weights $\mathbf{w}^*$,

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log P(y_i|\mathbf{x}_i, \mathbf{w}). \tag{2.35}$$

If we assume a Gaussian prior on y_i with standard deviation $\sigma_i = \sigma_j$ we get

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{f(\mathbf{x}_i,\mathbf{w})-y_i}{\sigma_i}\right)^2} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} (y_i - f(\mathbf{x}_i, \mathbf{w}))^2, \quad (2.36)$$

which is the classic L_2 loss suggested by Gauss. Similarly if we assume the Laplacian distribution, making measurements far from the mean a bit more likely, we get

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log \frac{1}{2b_i} e^{\left|\frac{f(\mathbf{x}_i,\mathbf{w})-y_i}{b_i}\right|} = \arg\min_{\mathbf{w}} \sum_{i=1}^{N} |(y_i - f(\mathbf{x}_i, \mathbf{w}))|, \quad (2.37)$$

which is the L_1 loss.

Classification

For classification we have a set of predetermined classes to predict. For example if an object is *background*, *cat* or *dog*. We encode this as predicting a set of integers, so we seek the function $f\colon \mathbb{R}^n \to \{1, .., k\}$ for k classes. From the last layer of the network we get a k dimensional vector of logits, $\hat{\mathbf{z}} \in \mathbb{R}^k$, where each element r represents how likely class r is. Since we would like to have the probability of each class, we use the Softmax operation to map our logits to $\hat{\mathbf{y}}$, where each element fulfil $\hat{y}^{(r)} \in [0, 1]$ and $\sum_r \hat{y}^{(r)} = 1$. The Softmax operation is defined as

$$\hat{y}^{(r)} = \frac{\exp(\hat{z}^{(r)})}{\sum_{s=1}^{k} \exp(\hat{z}^{(s)})}. \quad (2.38)$$

Now each element $\hat{y}^{(r)} \in \hat{\mathbf{y}}$ represents the likelihood of class r. For each input $\mathbf{x}_i$ we have a corresponding ground truth output y_i equal to the integer of the correct class. So in our example if image i depicts a *dog* then $y_i = 3$ since we ordered the classes as { *background*, *cat* or *dog* }. The maximum likelihood estimate for network weights $\mathbf{w}$ is

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} \sum_{i=1}^{N} \log \hat{y}_i^{(y_i)} \quad (2.39)$$

$$= \arg\min_{\mathbf{w}} -\sum_{i=1}^{N} \log \frac{\exp(\hat{z}_i^{(y_i)})}{\sum_{s=1}^{k} \exp(\hat{z}_i^{(s)})}, \quad (2.40)$$

which is known as the cross entropy loss. Since the classes in the training data often are unbalanced, for example twice as many dogs as cats, it is common to weight each term in the loss depending on the target class. If cats and dogs are equally likely to appear in our application domain then we will want an unbiased estimator. Then we get the common cross entropy loss formulation

$$L_{CE} = -\frac{1}{\sum_{i=1}^{N} \gamma^{(y_i)}} \sum_{i=1}^{N} \gamma_{y_i} \log \frac{\exp(\hat{z}_i^{(y_i)})}{\sum_{s=1}^{k} \exp(\hat{z}_i^{(s)})}, \tag{2.41}$$

where $\gamma^{(y_i)}$ is the loss weight for target class y_i.

Regularization

A good way to handle overfitting and improve generalization is to put some regularization on the network weights $\mathbf{w}$. Once again consider our model to be a probabilistic model and we instead of the ML estimate take the Maximum a Posteriori estimate (MAP). Then we get

$$\begin{aligned} \mathbf{w}^* &= \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D}) && (2.42) \\ &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w}), && (2.43) \end{aligned}$$

where $P(\mathbf{w})$ is our prior on the weights. Assuming a Gaussian prior with zero mean and standard deviation σ for all weights we get

$$\begin{aligned} \mathbf{w}^* &= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) - \sum_j \frac{w_j^2}{\sigma^2\sqrt{2}} && (2.44) \\ &= \arg\min_{\mathbf{w}} - \log P(\mathcal{D}|\mathbf{w}) + \beta\mathbf{w}^T\mathbf{w}, && (2.45) \end{aligned}$$

which is known as weight decay with design parameters β and puts an L_2 loss on the weights.

Chapter 3

Computer Vision - Cameras and Geometry

The research within the Computer Vision community spans a broad array of topics, many that will not be mentioned in this chapter. Most of the contributions in this thesis relates to Structure from Motion with sparse features, specifically points or lines. This chapter introduces related key concepts, starting with feature detection and camera models, then moving on to the mathematical 2D-to-3D relation and how a Structure from Motion pipeline is constructed. Finally the concept of map merging – studied in paper V-VII– for point clouds is given some background.

3.1 Image Features

Many Computer Vision applications require that we may mathematically relate one image to another. For example when tracking an object or to reconstruct a building from multiple views. To find relative pose between images it is common to use *features* in terms of geometrical primitives like points, lines, planes and ellipses. Each instance of said feature in the image is assigned a descriptor, which typically is a vector in $\mathbb{R}^N$, for some N. This descriptor hold information about the appearance of the feature so that it can be matched between images without any prior knowledge about the geometry of the cameras or the scene.

Some examples of point based engineered feature descriptors are SIFT [36], ORB [44], SURF [3] and BRIEF [8]. The modern Neural Network models have also brought learned point based feature descriptors, for example [13, 40, 49, 53]. For line detectors we have

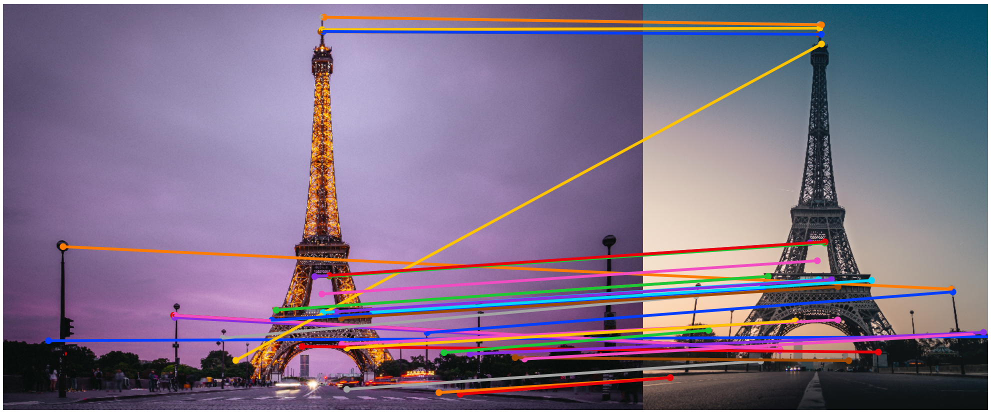


Figure 3.1: Two photos taken of the Eiffel Tower at different times from different locations. Keypoints (SIFT) are extracted from each image and matching by appearance is attempted. But due to the repeated structure from both poles and the building some matches are not correct.
Image credit: Denys Nevozhai and Gautier Salles from `unsplash.com`.

for example the engineered LSD [19] and EDlines [1]. Likewise there are learning based models – like LCNN [58], HAWP [52], Paper II – which are typically without descriptors and other examples like, SOLD2 [41] and DeepLSD [42] that are with descriptors.

Paper V-VII utilize point features to find corresponding points between images and construct a point cloud map. To find matching feature points between two images $\mathbf{X}$ and $\mathbf{Y}$ with keypoints and descriptors $(\mathbf{x}_i, \mathbf{f}_i) \in \mathbf{X}$ and $(\mathbf{y}_j, \mathbf{g}_j) \in \mathbf{Y}$ it is suggested by Lowe [36] to consider the relative distance to neighbours in the feature descriptor space. That two feature points $(\mathbf{x}_m, \mathbf{f}_m) \in \mathbf{X}$ and $(\mathbf{y}_n, \mathbf{g}_n) \in \mathbf{Y}$ match if

$$\frac{\min_{j, j \neq n} \|\mathbf{f}_m - \mathbf{g}_j\|}{\|\mathbf{f}_m - \mathbf{g}_n\|} < t, \tag{3.1}$$

for some threshold t. Common choices are within the range $t \in [0.6, 0.8]$. In essence, this requires the match to be significantly better than any other, which reduces the risk of false matches.

An example of such a matching of SIFT-features is shown in Figure 3.1. We see that the appearance-based matching does not yield a perfect result, there are some false matches. In Structure from Motion these are typically called outliers and are suppressed using the geometric constraints of the problem.

3.2 Camera Model

To calculate 3D geometry from images we need a mathematical model for the camera. There are different models available [21] but the most common approach is to use the

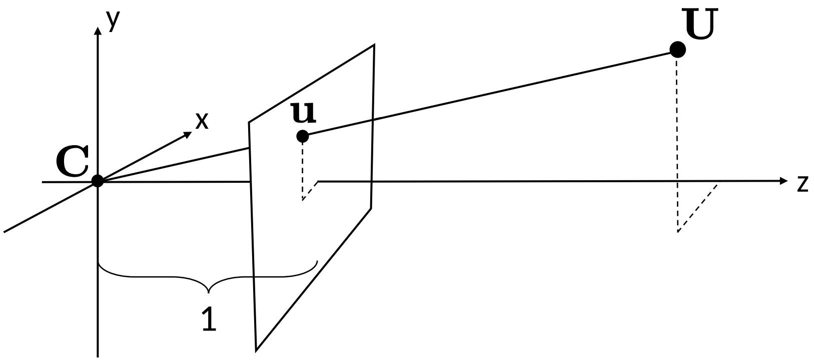


Figure 3.2: The pinhole camera model where $\mathbf{u}$ is a projection of $\mathbf{U}$ in the image plane at $z = 1$.
Image credit: Flood [17].

pinhole camera model coupled with a non-linear function to model lens distortion.

Let $\hat{\mathbf{U}} = [X, Y, Z] \in \mathbb{R}^3$ be a point in the 3D world and $\hat{\mathbf{u}} = [x, y] \in \mathbb{R}^2$ the projected pixel coordinate in the camera image. We get the projection $\hat{\mathbf{u}}$ by drawing a line from $\hat{\mathbf{U}}$ to the origin $\mathbf{C} = [0, 0, 0]$ and taking the intersection point at the image plane placed at $Z = 1$, i.e

$$\hat{\mathbf{u}} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}, \tag{3.2}$$

which is illustrated in Figure 3.2. To model the pose of the camera in relation to the reference world with origin $\mathbf{O}$ the coordinates are rotated and translated to the camera coordinate system by taking $\mathbf{R}\hat{\mathbf{U}} + \mathbf{t}$. These parameters are called the *extrinsic parameters*, see Figure 3.3. To facilitate the use of algebraic tools, it is common rewrite the coordinates as *homogeneous coordinates* in projective space $\mathbb{P}$. We rewrite $\hat{\mathbf{U}}$ and $\hat{\mathbf{u}}$ as

$$\mathbf{U} = \begin{bmatrix} \hat{\mathbf{U}} \\ 1 \end{bmatrix} \in \mathbb{P}^3, \quad \mathbf{u} = \begin{bmatrix} \hat{\mathbf{u}} \\ 1 \end{bmatrix} \in \mathbb{P}^2, \tag{3.3}$$

where $\mathbb{P}$ is the projective space for $\mathbb{R}$. Using this coordinate representation we can express the *finite projection camera model* used in the thesis as

$$\lambda\mathbf{u} = \mathbf{P}\mathbf{U} \quad \Leftrightarrow \quad \lambda\mathbf{u} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{U}, \tag{3.4}$$

where $\mathbf{P}$ is the *camera matrix*, $\lambda \in \mathbb{R}$ and

$$\mathbf{K} = \begin{bmatrix} \mu f & s & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

is the *intrinsic parameter* matrix. Here f is the *focal length*, (x_0, y_0) is the *principal point*, s is a *skew factor* and μ is the *aspect ratio*.

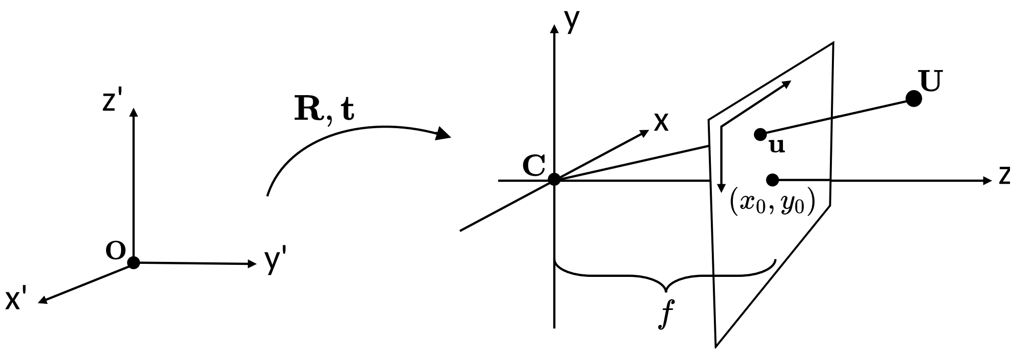


Figure 3.3: Shows how the finite projection model extends the pinhole camera by adding pose $\mathbf{R}$, $\mathbf{t}$, principal point (x_0, y_0) and focal length f.
Image credit: Flood [17].

3.3 Epipolar Geometry – Uncalibrated

Consider two cameras with camera matrices $\mathbf{P}$ and $\bar{\mathbf{P}}$ which observe a scene point $\mathbf{U}$. If neither camera pose nor scene point position is known, it is not possible to find an absolute estimate of either, the best we can do is a relative estimate. In fact, if the cameras are uncalibrated – $\mathbf{P}$ and $\bar{\mathbf{P}}$ are any projective transforms – the solution can only be determined up to a projective transform $\mathbf{H}$. By looking at the camera equation

$$\lambda \mathbf{u} = \mathbf{P}\mathbf{U} \quad \Leftrightarrow \quad \lambda \mathbf{u} = \mathbf{P}\mathbf{H}\mathbf{H}^{-1}\mathbf{U} = \lambda \mathbf{u} = \mathbf{P}^*\mathbf{U}^*, \tag{3.6}$$

it is clear that the equation holds for any projective matrix $\mathbf{H}$, but alters the camera pose and scene point.

Due to the ambiguity it is practical to use a relative parametrization of the problem by excluding the scene point. Without loss of generalization we may apply a projective transform placing one camera in the origin, i.e.

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}, \quad \bar{\mathbf{P}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \end{bmatrix}, \tag{3.7}$$

where $\mathbf{A} \in \mathbb{R}^{3\times 3}$. Let point $\mathbf{u}$ in the first image be a projection of the scene point $\mathbf{U}$, then $\mathbf{U}$ lies on a line drawn from the origin $\mathbf{C}$ and the projection point $\mathbf{u}$ as illustrated in Figure 3.4. Hence

$$\mathbf{U}(s) = \begin{bmatrix} \mathbf{u} \\ s \end{bmatrix}, \tag{3.8}$$

where $s \in \mathbb{R}$ governs where on the line $\mathbf{U}$ is. The projection of this line into the second camera is called the epipolar line $\bar{\mathbf{l}}$, each point given by

$$\bar{\mathbf{P}}\mathbf{U}(s) = \begin{bmatrix} \mathbf{A} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ s \end{bmatrix} = \mathbf{A}\mathbf{u} + s\,\mathbf{t}. \tag{3.9}$$

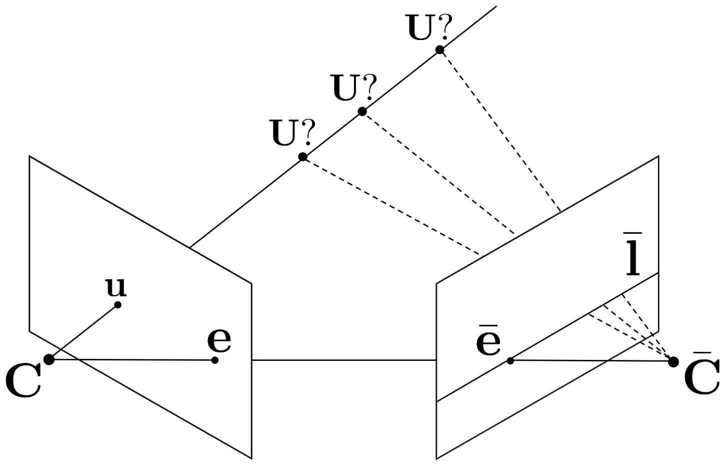


Figure 3.4: Illustration of the epipolar geometry model. Given the projected point $\mathbf{u}$ the projection into the second camera with center $\bar{\mathbf{C}}$ must be on the epipolar line $\bar{\mathbf{l}}$. Furthermore the epipoles $\mathbf{e}$ and $\bar{\mathbf{e}}$ lies on the line between the camera centers.
Image credit: Flood [17].

Specifically the projection of camera center $\mathbf{C}$ is known as the epipolar point $\bar{\mathbf{e}} = \bar{\mathbf{P}}\mathbf{C}$ and will always lie on the epipolar line $\bar{\mathbf{l}}$. The epipolar line may be written as $\bar{\mathbf{l}} = \bar{\mathbf{e}} \times \mathbf{A}\mathbf{u}$ in $\mathbb{P}^2$ and since $\bar{\mathbf{u}}$ will lie on this line the constraint is that

$$\bar{\mathbf{u}}^T\bar{\mathbf{l}} = \bar{\mathbf{u}}^T[\bar{\mathbf{e}}]_x\mathbf{A}\mathbf{u} = \bar{\mathbf{u}}^T\mathbf{F}\mathbf{u} = 0, \tag{3.10}$$

where $\mathbf{F}$ is called the fundamental matrix and $[\bar{\mathbf{e}}]_x$ denotes the matrix which by multiplication is equivalent to a cross product with $\bar{\mathbf{e}}$. Using 8 corresponding points between 2 images it is possible to solve for $\mathbf{F}$, this is known as the eight point algorithm [20, 35].

3.4 Epipolar Geometry – Calibrated

For the calibrated case – where $\mathbf{K}$ is known – each camera matrix has the form

$$\mathbf{P} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} \tag{3.11}$$

and the solution is determined up to a similarly transform with rotation, translation and scaling. Thus yielding $3 + 3 + 1 = 7$ degrees of freedom in the solution. Without loss of generality we write the two camera matrices as

$$\mathbf{P} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}, \quad \bar{\mathbf{P}} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}, \tag{3.12}$$

where $\mathbf{R}$ is a rotation matrix which is orthogonal with $\det(\mathbf{R}) = 1$. For the calibrated case the analog to the fundamental matrix is the essential matrix $\mathbf{E}$ which satisfies

$$\bar{\mathbf{x}}^T [\bar{\mathbf{t}}]_x \mathbf{R}\mathbf{x} = \bar{\mathbf{x}}^T \mathbf{E}\mathbf{x} = 0, \tag{3.13}$$

where

$$\bar{\mathbf{x}} = \mathbf{K}^{-1}\bar{\mathbf{u}}, \quad \mathbf{x} = \mathbf{K}^{-1}\mathbf{u}, \tag{3.14}$$

are the calibrated image coordinates. Since $\mathbf{E} = [\bar{\mathbf{t}}]_x \mathbf{R}$ it has $3 + 3 - 1 = 5$ DoF, because the translation and rotation matrix has 3 DoF each and scale is arbitrary. An effective and common method for estimating the essential matrix is the 5 point method [39]. Except for Equation (3.13) the additional constraints are

$$\det(\mathbf{E}) = 0, \tag{3.15}$$

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}\text{trace}(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0, \tag{3.16}$$

which are non-linear and cannot be solved with Direct Linear Transformation (DLT) as the eight point algorithm. However, by solving the DLT for 5 points we get a basis for the solution consisting of 4 basis matrices

$$\mathbf{E} = \gamma_1\mathbf{E}_1 + \gamma_2\mathbf{E}_2 + \gamma_3\mathbf{E}_3 + \gamma_4\mathbf{E}_4. \tag{3.17}$$

Since scale is arbitrary one coefficient may be fixed, for example be letting $\gamma_1 = 1$. The remaining 3 may be solved for by inserting the basis parametrization into non-linear constraints, forming polynomial equations in $\gamma_2, \gamma_3, \gamma_4$. These equations may for example be solved using a minimal solver, described in Section 3.6.

3.5 RANSAC

After a brief introduction to feature points and epipolar geometry it is clear that the theory is there to find a relative pose between 2 cameras given at least 5 corresponding feature points. Unfortunately the matches given by the appearance based feature descriptors are not always correct. There may be matches that are very wrong, for example in Figure 3.1 and will yield incorrect poses for the cameras. These matches are called *outliers* and to get a good estimate it is necessary to use robust methods to find and suppress them. Naturally, the matches that are correct are called *inliers*.

RANdom SAmple Consesus (RANSAC) [15] is a common method for robust model estimation. The idea is to find a small set of data points – in our case corresponding point pairs – that are inliers and use them to estimate a model. That model can then be used for outlier hypothesis testing. In Figure 3.5 the problem is illustrated for a line fitting problem. Given

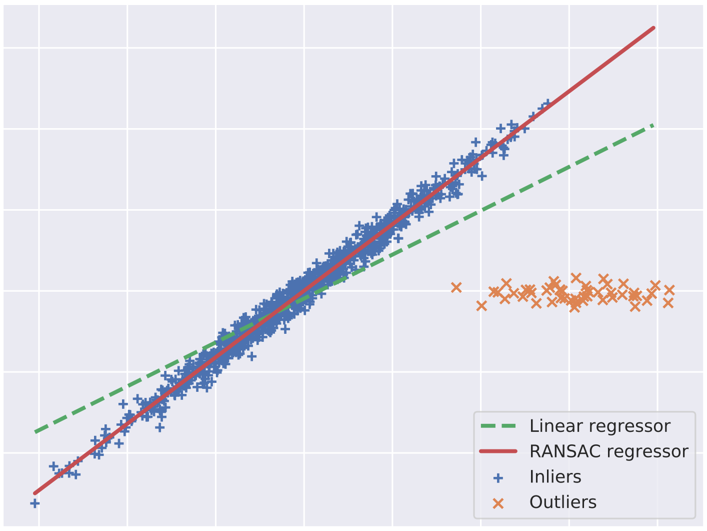


Figure 3.5: Illustrates the difference between using a standard linear regression and a RANSAC regressor in line fitting. Due to the large amount of outliers that does not follow the statistical model for the linear regressor, the green line does not fit any of the clusters. The RANSAC regressor finds the largest set of points that are consistent with the statistical model of the linear regressor and use that to fit the line.

data points that contains both inliers ($+$) and outliers ($\times$), we need to fit the best line. If all points are used in the regression of the line parameters, the line will go between the outliers and the inliers. While if RANSAC is used the model with best consensus is found using the inlier points to regress the line parameters. See Algorithm 3.1 for an outline of the algorithm. To successfully use RANSAC the most important factors are, *(i)* to choose the minimal sample size since a smaller sample size increases the probability to find a sample set with only inliers; *(ii)* to have a fast solver to regress the model from the sample set, since this solver is used for every iteration. Since this is a core part of SfM, where thousands of images need to be matched, a lot of research has gone into finding the minimal cases for estimating camera poses under different conditions and finding fast solvers for this. These are called minimal solvers and will be introduced in the next section.

3.6 Minimal Solvers

Since robust matching of images is essential in a SfM pipeline, it is important to develop fast *minimal solvers* to be used in the RANSAC iterations. A minimal solver is an algorithm

Algorithm 3.1: Principle of a RANSAC algorithm. Each iteration regresses a model from a minimal sample set. The idea is that for some iteration the sample set will be exclusively inliers and therefore that model will align with a lot of data points and get a high inlier count.

```
   Data: Data Points : $\mathbf{D}$, Sample size : $T$, Iterations: $K$
   Result: Model $M^*$ regressed from the largest inlier set $C^*$
1  $C^* \leftarrow \emptyset$
2  for $k = 1, ..., K$ do
3      $S \leftarrow$ samplePoints($\mathbf{D}$, $T$)
4      $M \leftarrow$ estimateModel($S$)
5      $C \leftarrow$ calculateInliers($M$, $\mathbf{D}$)
6      if size($C$) > size($C^*$) then
7          $C^* \leftarrow C$
8      end
9  end
10 $M^* \leftarrow$ estimateModel($C^*$)
```

designed to solve a specific *minimal problem*, which is a set of polynomial equations where the effective degree of freedom is equal to the number of variables to solve for. In this work the papers IV and VII have minimal solvers as contributions and the SfM pipeline in paper V and VI also utilize minimal solvers. Therefore an overview of presented in this section. For more extensive theory, see [9, 10, 32].

Let $\mathbf{x} = \{x_1, x_2, ..., x_m\}$ denote a set of variables and $\boldsymbol{\alpha} = \{\alpha_1, ...\alpha_m\}$ denote a multi-index vector, then

$$\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} x_2^{\alpha_2} \dots x_m^{\alpha_m} \tag{3.18}$$

is called a *monomial* in $\mathbf{x}$ of degree $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + ... + \alpha_m$. Furthermore, a polynomial may now be expressed as a linear combination of monomials

$$f(\mathbf{x}) = \sum_{\boldsymbol{\alpha}} c_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}, \quad c_{\boldsymbol{\alpha}} \in \mathbb{K}, \tag{3.19}$$

given that the sum is finite. Here $\mathbb{K}$ is a field. With this notation, a system of equations can be written as

$$\begin{cases} f_1(\mathbf{x}) & = 0 \\ f_2(\mathbf{x}) & = 0 \\ \quad\vdots & \\ f_l(\mathbf{x}) & = 0. \end{cases} \tag{3.20}$$

Since the problems in general are non-linear and multivariate they are difficult to solve fast. Methods for solving them can be divided into two categories [32], iterative methods and algebraic methods. However, most of the iterative methods are too slow to be used in a RANSAC framework in a SfM pipeline. In this thesis we use the automatic generator of *elimination templates* by Larsson *et al.* [33], which is based on a *Groebner basis* [9] approach.

The solver takes the structure of the system of Equations (3.20), in fact replaces the coefficients with random integers, and computes a Groebner basis which is utilized to find an elimination template. The elimination template is then used to find the corresponding *action matrix* [10] for the equations. By finding the eigenvalues of the action matrix, the solutions to the system of Equations (3.20) can be recovered.

The benefit of this approach is that once the elimination template is found, it can be used to – in practice – solve any other system of equations with the same structure, i.e. with different coefficients. For example the 5-point relative pose problem always produce the same structure, but the corresponding points change the coefficients of the terms. Since the elimination template can be optimized offline it is possible to build extremely fast solvers.

3.7 The Levenberg–Marquardt Algorithm

Both the typical SfM pipeline and the map merging papers solve the non-linear least squares problem where the error function may be written as

$$\epsilon(\mathbf{x}, \boldsymbol{\theta}) = \|\mathbf{x} - \hat{\mathbf{x}}(\boldsymbol{\theta})\|_2^2 = \sum_i r_i(\boldsymbol{\theta})^2 = \mathbf{r}^T\mathbf{r}, \tag{3.21}$$

where $\mathbf{x}$ are measurements and $\boldsymbol{\theta}$ parameters to be estimated. From the parameters a model is used to compute $\hat{\mathbf{x}}(\boldsymbol{\theta})$ that should be close to $\mathbf{x}$. The difference between one estimate and one measurement, i.e. the residual $r_i(\boldsymbol{\theta})$, then depends on the parameters. To simplify notation, we denote the residual vector $\mathbf{r} := \mathbf{r}(\boldsymbol{\theta})$.

Since there is no closed form solution we look to iterative optimization methods. The *Levenberg–Marquardt Algorithm* [16, 37] briefly introduced here is one of them. Using the step $\Delta\boldsymbol{\theta}$ we may approximate the error function using a Taylor expansion $\epsilon(\mathbf{x}, \boldsymbol{\theta} + \Delta\boldsymbol{\theta}) \approx \epsilon(\mathbf{x}, \boldsymbol{\theta}) + \mathbf{J}\Delta\boldsymbol{\theta}$, where $\mathbf{J} = \partial\epsilon/\partial\boldsymbol{\theta}$ is the *Jacobian*. Hence, each iteration is simplified to solving

$$\min_{\Delta\boldsymbol{\theta}} \|\epsilon + \mathbf{J}\Delta\boldsymbol{\theta}\|_2^2, \tag{3.22}$$

to find the step $\Delta\boldsymbol{\theta}$ in parameter space. Equation (3.22) is a linear least squares problem and can be solved in closed form by solving the *normal equations*

$$\mathbf{J}^T\mathbf{J}\Delta\boldsymbol{\theta} = \mathbf{J}^T\epsilon, \tag{3.23}$$

which results in the *Gauss-Newton method* [6]. The issue with this method is that the update is not guaranteed to reduce the error, since it is based on the Taylor expansion around $\boldsymbol{\theta}$. If the step is too large the approximation is no longer good. The Levenberg–Marquardt Algorithm mitigates the issue by solving the *augmented normal equations*

$$\left(\mathbf{J}^T\mathbf{J} + \mu\mathbf{I}\right)\Delta\boldsymbol{\theta} = \mathbf{J}^T\epsilon, \tag{3.24}$$

where $\mu > 0$ is a dampening factor that may be tuned to ensure convergence.

3.8 Structure from Motion

In Structure from Motion the task is to estimate the pose for each camera and each key object observed in the images. For papers V-VII the objects are keypoints with feature descriptors while in Paper IV they are parallel lines without descriptors. For this introduction we consider the case of keypoints. We have so far only considered solving relative pose for pairs of cameras. To solve the SfM problem there are two additions needed; *(i)* the method must optimize pose and 3D point positions over many images and *(ii)* the error metric used for finding the essential matrix is must be adjusted, since it is an algebraic error and not rooted in the geometry.

There are many algorithms for solving the SfM problem, many of which take a sequential approach, where pose and 3D points are first solved for an initial pair of cameras. This initial solution is then extended one camera at a time by camera resectioning, i.e. projecting the solved 3D points into the candidate camera image. New 3D points can then be added by triangulating corresponding points between the new camera and previously added cameras. While simple in the approach the method is sensitive to the quality of the initial solution and accumulating errors (drift). Therefore it is common to add a global optimization method called *bundle adjustment*, which iteratively optimize the *reprojection error*

$$\sum_{i,j} \left\| \mathbf{u}_{i,j} - \frac{\mathbf{P}_i\mathbf{U}_j}{\lambda_{i,j}} \right\|^2, \tag{3.25}$$

for all camera matrices $\mathbf{P}_i$ and all 3D points $\mathbf{U}_j$. The reprojection error is the squared distance between the projected 3D point $\mathbf{U}_j$ and the observed 2D keypoint $\mathbf{u}_{i,j}$ in camera with index i in the image plane. This error function results in a non-linear least squares problem, which may be solved using the Levenberg–Marquardt introduced in the previous Section 3.7.

Examples of popular sequential SfM algorithms are COLMAP[46] and Bundler[50]. There are also non-sequential approaches, for example [14], which are more robust since they do not rely as heavily on the initial camera pair.

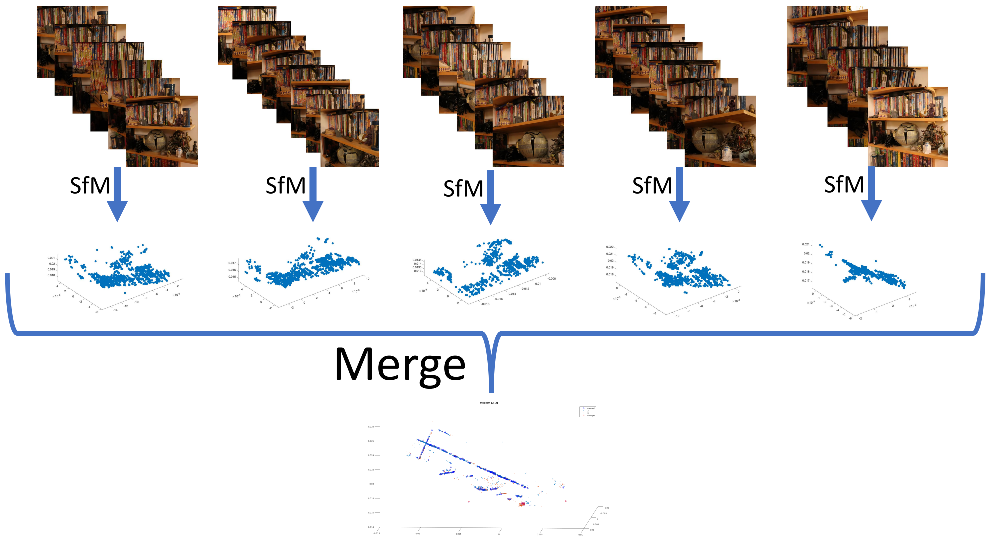


Figure 3.6: Map merging takes a set of overlapping point clouds from e.g. a SfM pipeline and merge them to a single point cloud, while minimizing an approximation of the reprojection error for each SfM solution.

3.9 Map Merging

Papers V-VII consider the topic of map merging, where a map consists of a 3D point cloud. As illustrated in Figure 3.6 a set of individual point clouds are merged to a single point cloud. For this introduction we assume that the point cloud is the result of a SfM solution and therefore is optimized with regard to the feature point reprojection error. Two naive approaches to merging would be to *(i)* take all images and perform a new bundle adjustment or *(ii)* assume the maps to be perfect and align them rigidly while optimizing on a spatial distance metric. The first approach is slow, but should be optimal. The second approach is fast but does not consider the uncertainty in the estimated maps with respect to the reprojection error.

This section will first describe a standard method for point cloud registration, which is option *(ii)*. Then follows an introduction to map merging, leaving the details to the papers.

3.9.1 Point Cloud Registration

Given two sets of point clouds, $\mathcal{P}_1 = \{\mathbf{p}_1, ..., \mathbf{p}_m\}$ and $\mathcal{P}_2 = \{\mathbf{q}_1, ..., \mathbf{q}_n\}$, with $\mathbf{p}_k, \mathbf{q}_l \in \mathbb{R}^3$, the problem of Point Cloud Registration is to find the optimal transform between the points clouds. The transform should place the point clouds as close as possible to each other and may be performed with or without known point correspondences.

A common method for registration without known point correspondences is the *Iterative Closest Point* (ICP) algorithm [4]. Essentially it is an iterative method where correspondence is assigned at every iteration based on the distance between points. After the assignment, registration is done using the hypothesized association and the iterations continue until convergence.

For known point correspondences $(\mathbf{p}_{k_i}, \mathbf{q}_{l_i}), i = 1, ..., N$ we want to find the similarity transform consisting of scale $s \in \mathbb{R}$, translation $\mathbf{t} \in \mathbb{R}^3$ and rotation matrix $\mathbf{R} \in SO^3$ that minimizes the distance between the corresponding points in Euclidean distance, i.e. we minimize

$$\sum_{i=1}^{N} \|\mathbf{p}_{k_i} - (s\mathbf{R}\mathbf{q}_{l_i} + \mathbf{t})\|^2. \tag{3.26}$$

The *Procrustes method* [23, 24, 47] solves this and is used for registration in Paper V and VII. The method first calculates the centroids

$$\bar{\mathbf{p}} = \frac{\sum_{i=1}^{N} \mathbf{p}_{k_i}}{N}, \quad \bar{\mathbf{q}} = \frac{\sum_{i=1}^{N} \mathbf{q}_{l_i}}{N}, \tag{3.27}$$

and then centers the point clouds

$$\hat{\mathbf{p}}_i = \mathbf{p}_{k_i} - \bar{\mathbf{p}}, \quad \hat{\mathbf{q}}_i = \mathbf{q}_{l_i} - \bar{\mathbf{q}}. \tag{3.28}$$

Assuming Gaussian measurement noise the covariance can be calculated as

$$\mathbf{H} = \sum_{i=1}^{N} \hat{\mathbf{p}}_i \hat{\mathbf{q}}_i^T, \tag{3.29}$$

and it is shown that the optimal rotation matrix $\mathbf{R}$ is the closest orthogonal matrix to $\mathbf{H}$. Therefore we can utilize the SVD decomposition $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$ and form the rotation matrix as

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T. \tag{3.30}$$

This may however yield a reflection, hence to make sure that $\det(\mathbf{R}) = 1$ it is calculated as

$$\mathbf{R} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U}\mathbf{V}^T) \end{bmatrix} \mathbf{V}^T. \tag{3.31}$$

Finally, the scale factor s and translation vector $\mathbf{t}$ are calculated as

$$s = \frac{\sum_{i=1}^{N} \hat{\mathbf{p}}_i^T \mathbf{R} \hat{\mathbf{q}}_i}{\sum_{i=1}^{N} \|\hat{\mathbf{p}}_i\|^2}, \quad \mathbf{t} = \bar{\mathbf{p}} - s\mathbf{R}\bar{\mathbf{q}}. \tag{3.32}$$

Since we seek a similarly transform there are seven degrees of freedom (1 scale, 3 translation, 3 rotation) and each point yields three constraints. Therefore it is necessary to have three point correspondences to solve for the transform.

3.9.2 Merging Point Clouds

As mentioned in the introduction of Section 3.9, to actually merge point clouds it is not enough to consider them static and find a geometric transform. The individual maps are not perfect and therefore there will be duplicates after the registration, where corresponding points are not perfectly aligned. Neither is it always feasible to perform a joint optimization over all the input data that the point clouds are based upon.

The middle ground is where merging methods, as described in Paper V-VII, are interesting solutions. Map merging has been studied previously in applications for loop closure and collaborative SLAM [2, 54, 57]. In most cases the focus of the research is on other parts of the system, rather than the point cloud merge. Recent reviews on map merging for multi-robot systems [2, 7] conclude that more research is needed on the merging parts.

The papers in this thesis that focus on map merging aims at finding an efficient method of merging, that does not require any of the input data used to estimate the individual point clouds. To this end the parameters are divided into *main parameters* and *auxiliary parameters*. Where the main parameters are the focus of the merge. In a SfM solution the main parameters could for example be all points that occurs in two or more maps, alternatively we might know that some points are static and choose these as the main parameters. The parameters related to the excluded points and camera matrices are handled as auxiliary parameters and their contribution to the error function is indirectly computed using Taylor approximation. This makes the merging efficient to compute. Furthermore, the estimate is grounded in the reprojection error and allows the maps to deform during merge according to the uncertainties in the individual map estimates.

Chapter 4

Research Summary and Conclusions

This work touches upon several research subjects for 2D to 3D reconstruction within Computer Vision and consists of seven papers. To conclude, this chapter will summarize the papers and their main contributions. The chapter is split into four separate sections, based on the research topics. Paper I evaluates the gain of using Bayesian Neural Networks in 3D reconstruction from singe view RGBD images and has its own conclusion section. Paper II-III consider new methods for detecting points, lines and planes in RGB images. These can be used as key objects in a SfM method, for example as developed in Paper IV, where line detections from parallel cylinders are used to solve the SfM problem. Finally the topic of map merging for SfM solutions is studied in Paper V-VII and also has its own section.

Paper I: In Depth Bayesian Semantic Scene Completion

This paper considers the task of Semantic Scene Completion (SSC) and the benefits of using a Bayesian Neural Network (BNN) for the task. The SSC task aims to re-construct the 3D geometry in a cone in front of the camera. The input given is an RGBD image

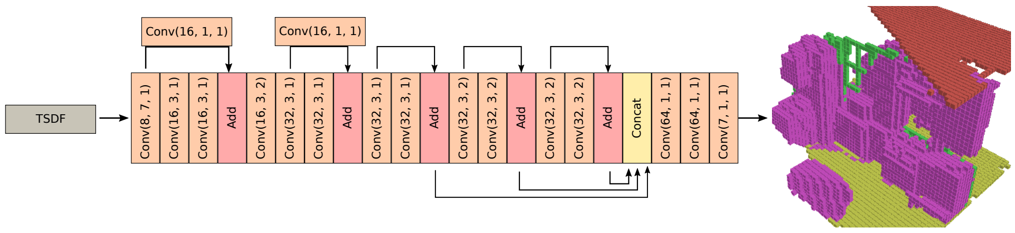


Figure 4.1: The model for Semantic Scene Completion in Paper I takes a TSDF representation of the RGBD image and predicts a labeled 3D voxel grid.

which consists of a color image (RGB) and a depth image (D), which typically is measured by round trip time of IR (Infra Red) light impulses with some form of LiDAR sensor. We assume that the intrinsic and extrinsic camera parameters are known so the we may map each image pixel to a 3D coordinate using the measured depth. The goal is to label a 3D voxel grid in front of the camera from a fixed set of semantic classes as described in [51].

A common approach to volumetric reconstruction from RGBD is to use a modification of the Signed Distance Function (SDF) [11], namely the Truncated SDF (TSDF) [28, 38], to map the continuous depth measurement to the discrete 3D voxel representation. Each voxel then stored the distance d to the nearest surface, where the sign dictates on which side of the surface the voxel is. In our approach we use the flipped TSDF $d_{flipped} = \text{sign}(d)(d_{max} - d)$ according to [51]. The resulting flipped TSDF 3D voxel grid is then used as input to our Bayesian Neural Network model.

The first contribution is a pair of BNN models with baseline deterministic CNN models developed for the task, released online[1]. See Figure 4.1 for one of the architectures. Secondly the SSC task is extended to include neighbouring rooms within the viewing cone, the idea being that the BNN might be better equipped to handle these areas with very weak prior better than a deterministic CNN. Thirdly, experiments are conducted for both the SSC task and the image classification task, where one class of objects is withheld in the training data and later introduced at inference. These experiments show that the BNN models perform better in standard evaluation metrics, therefore generalizing better, while also being better calibrated and predicting lower scores for the unseen objects in general. Lastly the paper performs an ablation study, evaluating the use of different activation functions and prior distributions for the network weights in the BNN models.

The main benefit of this approach is in the added robustness to unseen input, while the major downside is that the BNN model requires twice the amount parameters as the deterministic CNN, resulting in longer training time and requires more GPU memory resources at inference. Here it may be possible to gain most of the BNN benefits by using a standard CNN as backbone and only implement the network heads as BNN models, thereby limiting the resource problem.

Paper II-III: Indoor Wireframe and Polygon detection

Reconstructing indoor environments from monocular camera alone may be troublesome in certain scenarios. Since the methods often rely on point features, homogeneous surfaces like single color walls or repeated textures like wallpapers are difficult to match correctly across images. The ideas behind Paper II-III is to detect lines and planes which give better

[1]`https://github.com/DavidGillsjo/bssc-net`

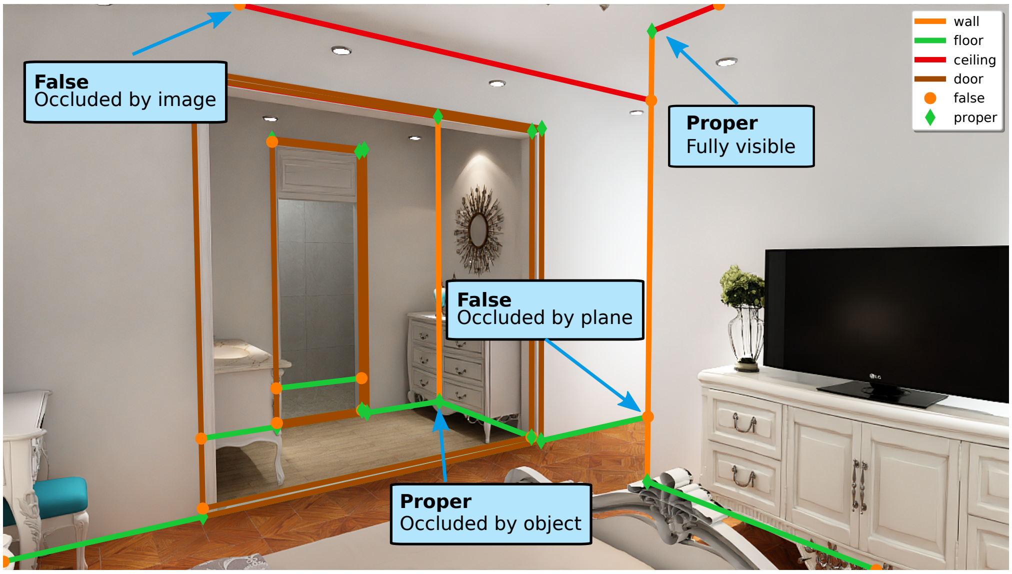


Figure 4.2: The proposed model in Paper II takes an image and outputs a semantic wireframe consisting of junctions and lines, where junctions are either *proper* or *false*. The line labels indicate which type of planes the line belong to.

geometric information about the room layout. These lines and planes may then be incorporated in a SfM and 3D reconstruction framework to increase robustness and performance. Both papers use synthetic images from digital room layouts in the dataset Structured3D [56].

Paper II: Semantic Room Wireframe Detection from a Single View

This paper defines a modified wireframe detection task, which takes the geometric representation of the wireframe as defined in previous work [27], but adapts it to represent the plane intersections of a room layout [22] . So that each line segment represents for example a floor-wall or wall-ceiling intersection, as illustrated in Figure 4.2. Furthermore we add semantic meaning to both the line segments and the junctions, which are endpoints of the line segments. Apart from these extensions to the task we also add windows and doors to the room layout and finally extend the annotation to neighbouring rooms that are visible through open doors. For this task a NN model is developed[2], which builds upon a previous wireframe detection model [52]. The model is extended with a Graph Convolutional Network (GCN) [30] module to improve performance. Furthermore the annotations for the defined task is made available online for the benefit of the research community.

While the output of this task is not as complete as the room layout estimation – which is a

[2]`https://github.com/DavidGillsjo/SRW-Net`

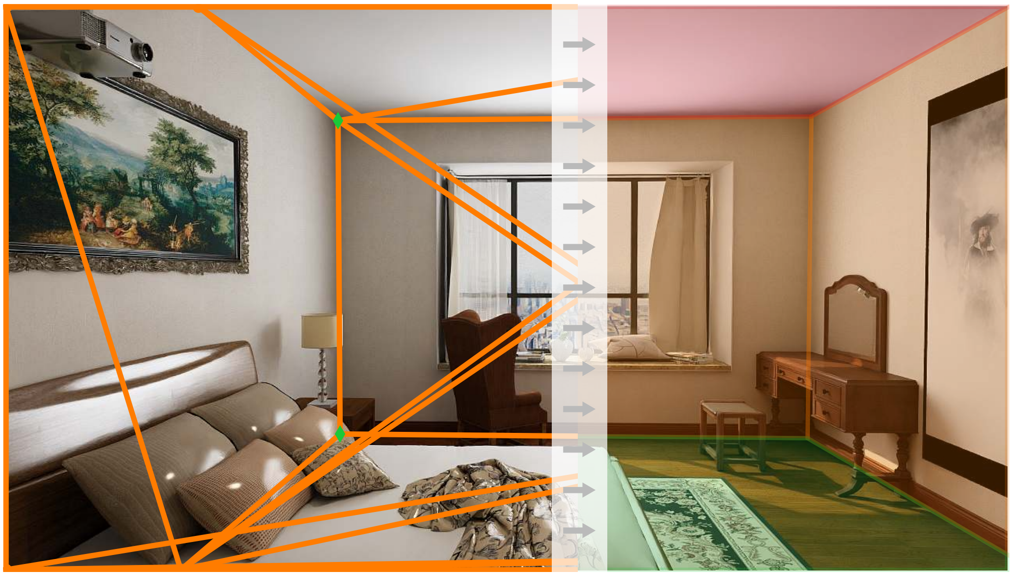


Figure 4.3: The proposed model in Paper III takes an image and a wireframe to produce a set of polygons corresponding to a Room Layout.

3D model of the layout – the major benefit is that it can handle complex layouts, e.g. detect a room wireframe through a doorway. It is also more flexible in the sense that it always give some information about the layout, since it may give partial output. If the output contains extra lines or missing lines, it can be accounted for in a multi-view SfM pipeline. A room layout model would have to estimate the full 3D model and may just skew the whole room to get a coherent structure.

Paper III: Polygon Detection for Room Layout Estimation using Heterogenous Graphs and Wireframes

This paper is in some sense a continuation, or complement, to Paper II. Instead of giving the line segments and junctions semantic classes, this model considers the planes and junctions as semantic objects. The data used is the same as for the standard room layout task, but with added semantic classes for planes and junctions. Since the model works with planes we evaluate and compare the model to room layout methods. The first contribution is a novel Heterogenous Graph Neural Network model[3] and a baseline polygon sampling model for estimating room layout from line segment detections, as illustrated in Figure 4.3. The second contribution is an evaluation of these models and comparison with a state of the art (SoTA) room layout estimation model. We show the potential of the model

[3]https://github.com/DavidGillsjo/polygon-HGT

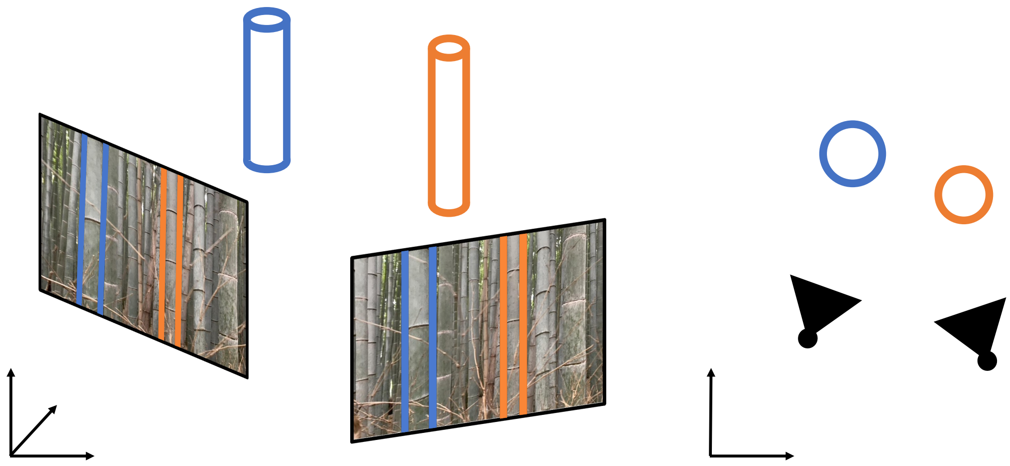


Figure 4.4: The theory developed in Paper IV considers a SfM pipeline for parallel cylinders. Which is equivalent to solving a 2D SfM problem with circles.

by simulating line detections and also show the performance while jointly predicting line detections. While only the simulated setup outperforms the SoTA model, it is to the best of our knowledge the first model to predict non-cuboid room layouts in 2D in an end-to-end fashion without heuristic post-processing. It is due to the Heterogenous Graph Transformers ability to reason globally over the polygons that overlapping polygons are avoided and proper room layouts are attained.

The current model is sensitive during training when jointly predicting a wireframe and room layout polygons. Which is, in our guess, the main reason that the performance is not as good. Future work would be to evaluate other loss terms to help the training process along. For example adding pseudo-tasks or using static samples from the ground truth wireframes. Due to the bottom-up nature in the approach it is sensitive to difficult occlusions around the wireframe components. Future work for addressing this could be to add a plane detector or to add a generative step where the model may add junctions and lines to the graph using link prediction.

Paper IV: The Multi-View Geometry of Parallel Cylinders

On the topic of using an alternative to points for SfM, this paper uses silhouettes for parallel cylinders in terms of two lines per cylinder, as illustrated in Figure 4.4. The main assumption is that all observed cylinders are parallel in 3D, then the problem may be reduced to a 2D reconstruction. The contribution is a study of the SfM problem for multiple views and multiple cylinders, including developing theory for the two-view relative motion and two-circle relative structure problem. We derive fast solvers for the minimal problems as well

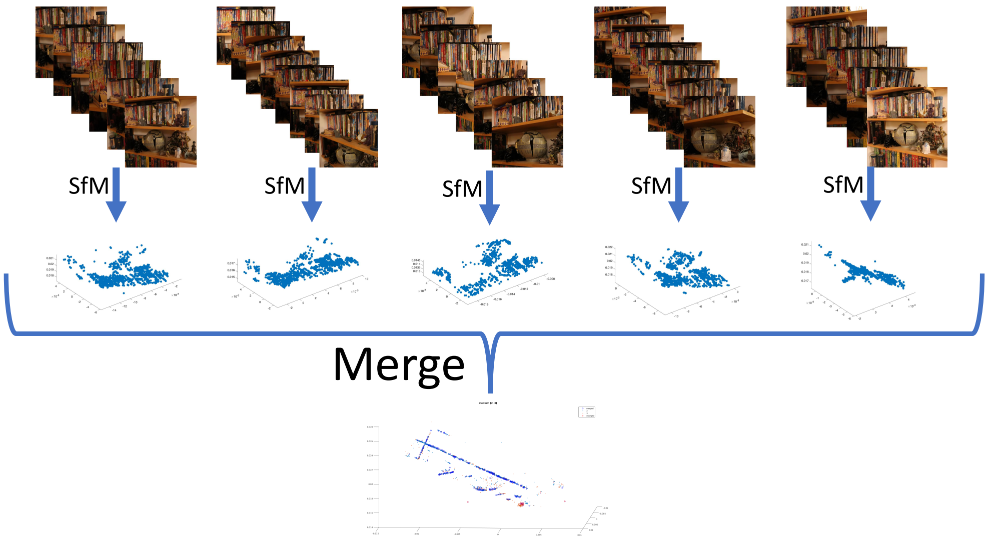


Figure 4.5: Map merging takes a set of overlapping point clouds from e.g. a SfM pipeline and merge them to a single point cloud, while minimizing an approximation of the reprojection error for each SfM solution.

as for the over-determined problem and use Sampson approximation for better accuracy in the geometric error of the solution. Experiments are conducted using a full SfM pipeline including bundle adjustment showing that the system works for both synthetic and real data.

For this paper the line features are manually annotated, but for future work it would be interesting to develop a fully automatic pipeline with a cylinder silhouette line detector – similar to the one proposed in Paper II. Furthermore it could be used to augment a keypoint based SfM pipeline to add robustness, since the cylinder shapes often are invariant to shifts in daytime and season.

Paper V-VII: Map merging

These three papers consider the problem of map merging, where a map is a point cloud, which could be generated by a SfM pipeline. As illustrated in Figure 4.5, Paper V-VI focus on the theory for effective merging of maps without the need to make a new global bundle adjustment. Paper VII considers minimal solvers which allow certain deformation of the point clouds to enable more robust matching between point clouds. Merging can then be done after the point correspondences are found.

Paper V: Efficient Merging of Maps and Detection of Changes

In this paper the problem of map merging is introduced in a Sender-Receiver and Structure from Motion setting. The main contribution is a method for efficient merging of maps optimized with bundle adjustment. The key idea is to factorize the Jacobian for each map by separating the joint and individual information into main parameters and auxiliary parameters, respectively. Taylor expansion is then used to get an approximation of how a change in the main parameters affect the auxiliary parameters. The map merging may then be expressed only in the main parameters and may be solved linearly if in the same coordinate system, which is a very beneficial. The second contribution is a method for change detection. Under the assumption that the individual maps have errors that follow a mean zero Gaussian distribution, we show how these errors together with the merged map error can be used to statistically determine if there has been a change in joint 3D points. For example if an object has moved or one of the individual maps are incorrect.

The method assumes that the individual maps are already aligned to the same coordinate system, which is not realistic since each SfM solution is only determined up to scale, rotation and translation. The alignment is done in a pre-processing step using the Procrustes method described in Section 3.9.1. Furthermore point correspondences between the individual maps are assumed to be given, which is not a trivial problem.

Paper VI: Generic Merging of Structure from Motion Maps with a Low Memory Footprint

The method for map merging is further developed in this paper, taking previous restrictions into consideration and formalizing the theory. The main contribution is a method for map merging, which in addition to Paper V also estimates the relative transform between the individual maps together with the 3D points and camera matrices. The transform between the maps are assumed to be a similarity transform which makes the merge process non-linear. Here we use a bundle adjustment procedure over the compressed residuals and registration transforms, though no longer linear it is still much faster than doing a full bundle adjustment over all points and cameras for the individual maps. The method is also extended to handle non-overlapping data, i.e. that some of the joint 3D points are missing in one or several individual maps. Finally, the change detection from Paper V is updated to work for the new method. Just like in the previous paper, point correspondences are given. Code for the map merging method is available online[4].

[4]https://github.com/gabrielleflood/mapmerging

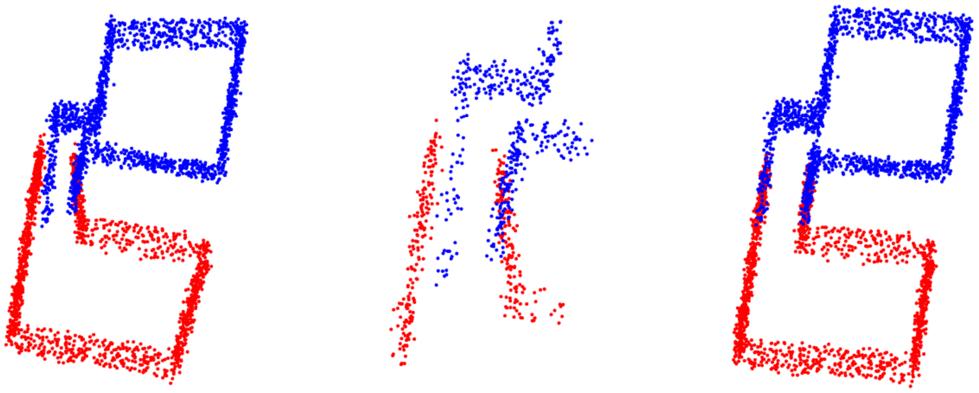


Figure 4.6: This is a motivating example, where two point clouds with drift should be merged. To the left and in the middle we see how a rigid matching would register the point clouds. Nt here the corridor does not align properly. To the right we illustrate the result of a method that allow deformation of the point clouds, resulting in a proper merge.

Paper VII: Minimal Solvers for Point Cloud Matching with Statistical Deformations

This paper considers the point correspondence problem for proper registration between the individual maps that are assumed to be given in Paper V and VI. The matching process is analogous to the image feature point matching process in the SfM pipeline. As input there is a set of tentative matches between the 3D point clouds, where an inlier set is found using a RANSAC procedure. The main contribution of the paper is the minimal solvers used in the RANSAC iterations that allow for some deformation of the point clouds. This is important since there may be drift in any individual map, as shown in Figure 4.6. The solvers considered are for the minimal registration problem with and without known scale. The number of modes for variation is also varied, resulting in 4 different solvers. The second contribution is a minimal solver for matching duplicate points withing maps, which may be used for loop closure. Experiments are conducted where the solvers are utilized in a RANSAC framework and the matches used for map merging according to the methods from Paper VI. They show that the solvers perform better than the Procrustes method for matching between maps with drift. The code for the solvers is available online[5].

[5]https://github.com/gabrielleflood/statistical-mapmatching-minsolv

References

[1] C. Akinlar and C. Topal. Edlines: Real-time line segment detection by edge drawing (ed). In *18th IEEE International Conference on Image Processing*, pages 2837–2840, 2011. doi: 10.1109/ICIP.2011.6116138.

[2] I. Andersone. Heterogeneous map merging: State of the art. *Robotics*, 8(3):74, 2019.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

[4] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. In *Sensor fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[5] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*, 2015.

[6] L.-C. Böiers. *Mathematical methods of optimization*. Studentlitteratur AB, 2010.

[7] A. Bokovoy, K. Muraviev, and K. Yakovlev. Map-merging algorithms for visual SLAM: Feasibility study and empirical evaluation. In *Russian Conference on Artificial Intelligence*, pages 46–60. Springer, 2020.

[8] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*, pages 778–792. Springer, 2010.

[9] D. A. Cox, J. B. Little, and D. O'Shea. Ideals, varieties, and algorithms: an introduction to computational algebraic geometry and commutative algebra. *Undergraduate Texts in Mathematics*, 1997.

[10] D. A. Cox, J. Little, and D. O'shea. *Using algebraic geometry*. Springer Science & Business Media, second edition edition, 2005.

[11] B. Curless and M. Levoy. A volumetric method for building complex models from range images. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 303–312, New York, NY, USA, 1996. Association for Computing Machinery. ISBN 0897917464. doi: 10.1145/237170.237269. URL https://doi.org/10.1145/237170.237269.

[12] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[13] D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperPoint: Self-supervised interest point detection and description. In *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 224–236, 2018.

[14] O. Enqvist, F. Kahl, and C. Olsson. Non-sequential structure from motion. In *IEEE International Conference on Computer Vision Workshops*, pages 264–271. IEEE, 2011.

[15] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[16] R. Fletcher. *A modified Marquardt subroutine for non-linear least squares*. AERE report / R.: AERE report. Atomic Energy Research Establishment, Harwell, 1971.

[17] Flood, Gabrielle. *Mapping and Merging Using Sound and Vision : Automatic Calibration and Map Fusion with Statistical Deformations*. PhD thesis, Lund University, 12 2021. URL https://lup.lub.lu.se/search/files/109788897/gflood_thesis_electronic.pdf.

[18] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

[19] R. Grompone von Gioi, J. Jakubowicz, J.-M. Morel, and G. Randall. Lsd: A fast line segment detector with a false detection control. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4):722–732, 2010. doi: 10.1109/TPAMI.2008.300.

[20] R. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997. doi: 10.1109/34.601246.

[21] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. second edition.

[22] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *IEEE 12th international conference on computer vision*, pages 1849–1856. IEEE, 2009. doi: 10.1109/ICCV.2009.5459411.

[23] B. K. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of Ameraca (JOSA) A*, 4(4):629–642, 1987.

[24] B. K. Horn, H. M. Hilden, and S. Negahdaripour. Closed-form solution of absolute orientation using orthonormal matrices. *Journal of the Optical Society of Ameraca (JOSA) A*, 5(7):1127–1135, 1988.

[25] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90020-8. URL https://www.sciencedirect.com/science/article/pii/0893608089900208.

[26] Z. Hu, Y. Dong, K. Wang, and Y. Sun. Heterogeneous graph transformer. In *Proceedings of The Web Conference*, WWW '20, page 2704–2710, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450370233. doi: 10.1145/3366423.3380027. URL https://doi.org/10.1145/3366423.3380027.

[27] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma. Learning to parse wireframes in images of man-made environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[28] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, page 559–568, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450307161. doi: 10.1145/2047196.2047270.

[29] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In Y. Bengio and Y. LeCun, editors, *ICLR (Poster)*, 2015.

[30] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[31] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. Burges, L. Bottou, and K. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[32] V. Larsson. *Computational Methods for Computer Vision: Minimal Solvers and Convex Relaxations*. Lund University, 2018.

[33] V. Larsson, K. Åström, and M. Oskarsson. Efficient solvers for minimal problems by syzygy-based reduction. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 820–829. IEEE, 2017.

[34] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Doll'a r, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[35] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, Sep 1981. ISSN 1476-4687. doi: 10.1038/293133a0. URL https://doi.org/10.1038/293133a0.

[36] D. G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[37] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[38] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon. Kinectfusion: Real-time dense surface mapping and tracking. In *10th IEEE International Symposium on Mixed and Augmented Reality*, pages 127–136, 2011. doi: 10.1109/ISMAR.2011.6092378.

[39] D. Nister. An efficient solution to the five-point relative pose problem. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages II–195. IEEE, 2003.

[40] Y. Ono, E. Trulls, P. Fua, and K. M. Yi. LF-Net: Learning local features from images. In *32nd International Conference on Neural Information Processing Systems (NIPS)*, pages 6237–6247, 2018.

[41] R. Pautrat*, J.-T. Lin*, V. Larsson, M. R. Oswald, and M. Pollefeys. Sold2: Self-supervised occlusion-aware line description and detection. In *Computer Vision and Pattern Recognition (CVPR)*, 2021.

[42] R. Pautrat, D. Barath, V. Larsson, M. R. Oswald, and M. Pollefeys. Deeplsd: Line segment detection and refinement with deep image gradients. In *Computer Vision and Pattern Recognition (CVPR)*, 2023.

[43] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets, 2022.

[44] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International Conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[45] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, Oct 1986. doi: 10.1038/323533a0. URL https://doi.org/10.1038/323533a0.

[46] J. L. Schonberger and J.-M. Frahm. Structure-from-motion revisited. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113. IEEE, 2016.

[47] P. H. Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[48] K. Shridhar, F. Laumann, and M. Liwicki. A comprehensive guide to bayesian convolutional neural network with variational inference, 2019.

[49] E. Simo-Serra, E. Trulls, L. Ferraz, I. Kokkinos, P. Fua, and F. Moreno-Noguer. Discriminative learning of deep convolutional feature point descriptors. In *IEEE International Conference on Computer Vision (ICCV)*, pages 118–126. IEEE, 2015.

[50] N. Snavely, S. M. Seitz, and R. Szeliski. Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2):189–210, 2008.

[51] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser. Semantic scene completion from a single depth image. *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[52] N. Xue, T. Wu, S. Bai, F.-D. Wang, G.-S. Xia, L. Zhang, and P. H. Torr. Holistically-attracted wireframe parsing. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[53] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. LIFT: Learned invariant feature transform. In *European Conference on Computer Vision (ECCV)*, pages 467–483. Springer, 2016.

[54] S. Yu, C. Fu, A. K. Gostar, and M. Hu. A review on map-merging methods for typical map types in multiple-ground-robot SLAM solutions. *Sensors*, 20(23):6988, 2020.

[55] S. Zhang, H. Tong, J. Xu, and R. Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):11, Nov 2019. ISSN 2197-4314. doi: 10.1186/s40649-019-0069-y. URL https://doi.org/10.1186/s40649-019-0069-y.

[56] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou. Structured3d: A large photo-realistic dataset for structured 3d modeling. In *Proceedings of The European Conference on Computer Vision (ECCV)*, 2020.

[57] X. S. Zhou and S. I. Roumeliotis. Multi-robot SLAM with unknown initial correspondence: The robot rendezvous case. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1785–1792. IEEE, 2006.

[58] Y. Zhou, H. Qi, and Y. Ma. End-to-end wireframe parsing. In *ICCV*, 2019.

Scientific Publications

Paper I

In Depth Bayesian Semantic Scene Completion

David Gillsjö and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: For autonomous agents moving around in our world, mapping of the environment is essential. This is their only perception of their surrounding, what is not measured is unknown. Humans have learned from experience what to expect in certain environments, for example in indoor offices or supermarkets.

This work studies Semantic Scene Completion which aims to predict a 3D semantic segmentation of our surroundings, even though some areas are occluded. For this we construct a Bayesian Convolutional Neural Network (BCNN), which is not only able to perform the segmentation, but also predict model uncertainty. This is an important feature not present in standard CNNs.

We show on the MNIST dataset that the Bayesian approach performs equal or better to the standard CNN when processing digits unseen in the training phase when looking at accuracy, precision and recall. With the added benefit of having better calibrated scores and the ability to express model uncertainty.

We then show results for the Semantic Scene Completion task where a category is introduced at test time on the SUNCG dataset. In this more complex task the Bayesian approach outperforms the standard CNN. Showing better Intersection over Union score and excels in Average Precision and separation scores.

Keywords: Scene Completion, Deep Learning, Bayesian Neural Network

1 Introduction

Semantic scene completion is a challenging task in which both visible and occluded surfaces are labeled semantically in 3D. The problem naturally arises when aiming to predict a 3D scene from a single view, but can also be studied for multiple views.

Predicting occluded areas can be of great help to autonomous vehicles during navigation and exploration. Especially for UAVs (Unmanned Aerial Vehicles), which navigate a 3D space where observations may be sparse, scene completion can be used for smoother trajectories during path planning. During exploration it can help the agent understand the likelihood of free space in occluded areas. In Figure 1 we see an illustration of the problem where a UAV would benefit from knowing what to expect in occluded areas.

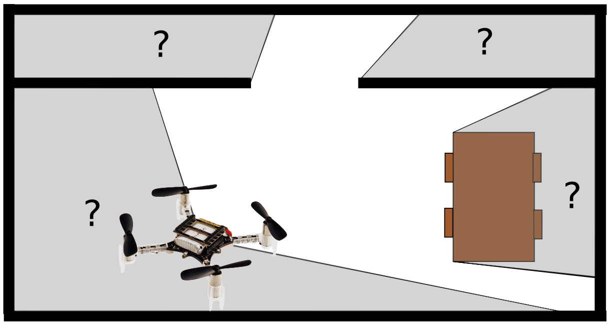


Figure 1: An UAV has some occluded areas in its surrounding and would like to have an idea about what to expect.

This work introduces a system for Bayesian Semantic Scene Completion (BSSC), which along with the prediction scores also delivers an estimation of uncertainty. This is crucial for decision making during autonomous navigation and exploration as it can help the agent understand when the data is new to the model and the prediction should not be trusted. It can also be used to understand what data should be added to the training to improve robustness.

We first test our implementation on the MNIST dataset to verify the Bayesian approach and to understand how the output distributions look for a well modeled dataset. We then move on to the SUNCG data for the Semantic Scene Completion task.

Our contributions include:

- An open source system for BSSC using Variational Inference released on `https://github.com/DavidGillsjo/bssc-net`.
- An extended SSC task on the SUNCG dataset which includes more occluded space.
- Experiments showing that the Bayesian approach is more robust to unseen data in the SSC task.
- Parameter studies on both MNIST and SUNCG.

2 Related work

Semantic scene completion has been formulated for both the single view and multiple view problem with different sensor modalities. One example is SSC-Net [1] that solves the single view using a 3D CNN with depth data as input. Our work is heavily inspired by their architecture and training setup. Other works [2, 3] have then extended the architecture by utilizing the RGB information as well.

Since the solutions are similar to Semantic Scene Segmentation, the following works are also of interest. Rather than feeding the whole input volume through the network, [4] takes a sliding window approach to better handle large datasets. By encoding the RGB information as features and using differentiable backprojection 3DMV [5] got impressive results on the ScanNet dataset [6].

The best performing networks on ScanNet semantic segmentation is currently based on point clouds [7–9] or sparse convolutions [10].

Our work will not focus on performance, but rather how to estimate uncertainty in semantic scene completion using Bayesian NN.

There are several ways of modeling a Bayesian Neural Network, this work uses the Bayes by Backpropagation [11]. Our implementation of Bayes by Backpropagation is based on [12] which implements this for a 2D CNN using softplus and normalized softplus as activation functions.

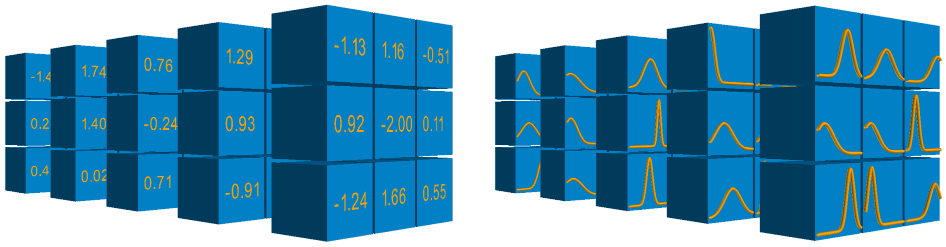
Both occupancy maps and semantic segmentation typically use a probabilistic representation where each class is assigned probability between 0 and 1. If the algorithm also outputs uncertainty together with the probability score, what does that mean and how do we use it? This is studied for Gaussian Process Occupancy Maps (GPOM) [13] in the robotics community. GPOM yields a mean and variance per grid cell, these are then fed to a linear classifier, which is trained to output a probability score.

Most NN classifiers does not have calibrated probabilities, this can be adjusted with for example Platt Scaling as in [14]. Uncertainties from Bayesian NN classifier can be similarly calibrated. In [15] the authors study calibration for different BNNs by comparing Entropy, Mutual Information, Aleatoric uncertainty and Epistemic uncertainty under different perturbations of input data. They conclude that predictive entropy and Epistemic uncertainty provide the most robust uncertainty estimates.

3 Bayes by backprop

The method introduced by [11] is based on Variational Inference. The main idea is to let each weight in the network be sampled from a distribution, where the distribution is learned at training time, as illustrated in Figure 2.

Let $P(\mathbf{w}|\mathcal{D})$ be the posterior for our Bayesian Neural Network. Given the posterior we



(a) Standard

(b) Bayesian

Figure 2: In 2a we see a filter bank from a standard 2D CNN, each weight is a scalar. In 2b we see a filter bank in a Bayesian Variational Inference 2D CNN, here each weight represented as a distribution which is sampled from at inference time.

can make predictions given unseen data by taking the expectation over the posterior

$$\begin{aligned} P(\hat{\mathbf{y}}|\hat{\mathbf{x}}) &= \mathbb{E}_{P(\mathbf{w}|\mathcal{D})}\left[P(\hat{\mathbf{y}}|\hat{\mathbf{x}},\mathbf{w})\right] \\ &= \int P(\hat{\mathbf{y}}|\hat{\mathbf{x}},\mathbf{w})P(\mathbf{w}|\mathcal{D})\,d\mathbf{w}. \end{aligned}$$

We estimate the posterior using a simpler model $q(\mathbf{w}|\theta)$ with learnable parameters θ, which minimizes the Kullback-Leibler (KL) divergence to the true posterior, i.e.

$$\begin{aligned} \theta^* &= \arg\min_{\theta} KL[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})] \\ &= \arg\min_{\theta} \int q(\mathbf{w}|\theta)\log\frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \\ &= \arg\min_{\theta} KL[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})]. \end{aligned}$$

This cost function is known as variational free energy or expected lower bound. We denote it as

$$\mathcal{F}(\mathcal{D},\theta) = KL[q(\mathbf{w}|\theta)||P(\mathbf{w})] - \mathbb{E}_{q(\mathbf{w}|\theta)}[\log P(\mathcal{D}|\mathbf{w})]. \tag{1}$$

Where the first part is the complexity cost as it enforces model simplicity using the prior. The second part is the likelihood, which describes how well the model describes the data.

Using Monte Carlo sampling the cost in (1) is approximated as

$$\begin{aligned} \mathcal{F}(\mathcal{D},\theta) \approx \sum_{i=1}^{n}\frac{\beta}{n}&\left[\log q(\mathbf{w}^{(i)}|\theta) - \log P(\mathbf{w}^{(i)})\right] \\ &- \log P(\mathcal{D}|\mathbf{w}^{(i)}), \end{aligned}$$

where $\mathbf{w}^{(i)}$ is a sample from the variational posterior $q(\mathbf{w}^{(i)}|\theta)$. This approximation enables more priors and posteriors since a closed for solution for the first term in (1) is not necessary. The scale factor $\frac{\beta}{n}$ with β as design parameter is introduced to tune the amount of regularization from the complexity cost.

3.1 Variational Posterior

Assuming the variational posterior $q(\mathbf{w}|\theta)$ is a diagonal Gaussian distribution. With the re-parametrization trick [16] the weights can be sampled from the posterior as

$$\mathbf{w} = \mu + \log(1 + e^{\gamma}) \odot \epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

where the model parameters are $\theta = (\mu, \gamma)$ and $\sigma = \log(1+e^{\gamma})$ to ensure that the standard deviation remains positive during optimization.

3.2 Prior

For the choice of prior we tested both Gaussian (as in [12]), Cauchy and a scale mixture of two Gaussian (as in [11]). The two latter affords more flexibility to the network and we found they yielded the best results. We mainly worked with Cauchy since it is more efficient to compute but have some results on the Gaussian scale mixture. As they are centered around 0 we denote them as

$$\text{Cauchy}(\gamma) = \frac{1}{\pi\gamma\left[1 + (\frac{x}{\gamma})^2\right]}$$

and

$$\text{G}^{M}_{(\sigma_0,\sigma_1)}(\alpha) = \alpha\mathcal{N}(0, \sigma_0^2) + (1-\alpha)\mathcal{N}(0, \sigma_1^2),$$

where $\mathcal{N}$ denotes the probability density function of the normal distribution.

3.3 Prediction & Uncertainty

The predictive mean and uncertainty is computed by sampling our variational posterior (BCNN) at test time. An unbiased estimation of the expectation is given [12] by

$$\begin{aligned}\mathbb{E}_{q(\mathbf{w}|\theta)}\left[P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w})\right] &= \int q(\mathbf{w}|\theta) P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w}^{(t)})\, d\mathbf{w} \\ &\approx \frac{1}{T}\sum_{t=1}^{T} P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w}^{(t)}),\end{aligned}$$

where $P(\hat{\mathbf{y}}|\hat{\mathbf{x}}, \mathbf{w}^{(t)})$ is the softmax output from forward pass t. To simplify notation we now denote this as $\mathbf{p}_t$.

There are a number of choices when it comes to measuring the uncertainty [15, 17]. Common choices are predictive entropy, aleatoric uncertainty and epistemic uncertainty.

Predictive Entropy - Measures the diversity of the distribution, so a higher entropy corresponds to higher uncertainty. It is computed as

$$H = -\sum_{t=1}^{T} \mathbf{p}_t \log \mathbf{p}_t.$$

Aleatoric Uncertainty - Corresponds to measurement noise from the input data, so increasing the size of the dataset should not reduce this uncertainty. It is computed as

$$\sigma_a^2 = \frac{1}{T}\sum_{t=1}^{T} \operatorname{diag}(\mathbf{p}_t) - \mathbf{p}_t\mathbf{p}_t^\top.$$

Epistemic Uncertainty - Corresponds to model uncertainty and will be low when input data is similar to training data. Increasing the amount and diversity of training data should reduce this uncertainty. This is computed as

$$\sigma_e^2 = \frac{1}{T}\sum_{t=1}^{T} (\mathbf{p}_t - \bar{\mathbf{p}}_t)(\mathbf{p}_t - \bar{\mathbf{p}}_t)^\top,$$

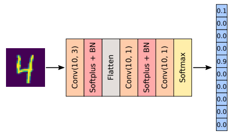
where $\bar{\mathbf{p}}_t = \frac{1}{T}\sum_{t=1}^{T} \mathbf{p}_t$ is the predictive mean from above.

4 Network architecture

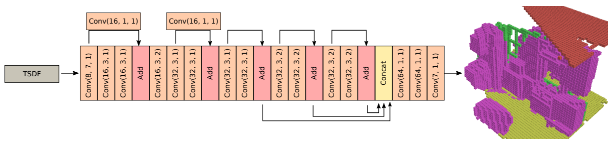
We have explored two network architectures. The first network architecture is inspired by [1], but we have included batch normalization [18] and used dilated convolutions instead of max pooling layers and strided convolutions. This to keep the resolution [19]. We denote it as SSC-Net. The second architecture is a UNet [20] with max pooling and transposed convolutions as up-sampling. Just as in [12] we chose softplus as activation functions instead of relu to have more active weights in the network. This spread is beneficial for a well calibrated uncertainty [15]. The architecture of the network is displayed in Figure 3.

5 Training

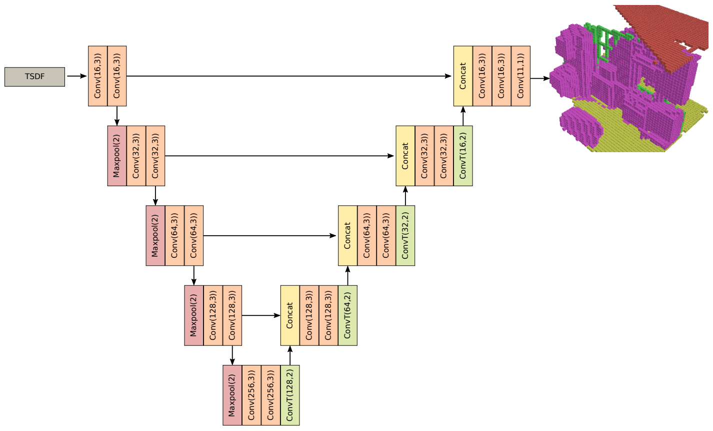
For training we use the Adam optimizer [21] with constant learning rate of 10^{-3}. We do not use weight decay since we regularize using KL divergence. For the weights we sample the mean uniformly as $\mu \sim U(-\frac{1}{\sqrt{k^d}}, \frac{1}{\sqrt{k^d}})$, where k is the kernel size in one dimension and



(a) MNIST



(b) SSC-Net



(c) UNet

Figure 3: Architecture of the BCNNs used for MNIST and SUNCG experiments. Conv(d, k, l) stands for a 3D convolution filter stack of depth d and kernel size k and dilation l. ConvT(d, k) is the up sampling operation Transposed Convolution with depth d, kernel and stride k. Batch normalization and softplus activation is performed after every Conv layer. Softmax in the final layer.

d is the number of dimensions. The parameter σ is simply initialized as a chosen constant σ_0.

As noted in [12] it is beneficial to set σ_0 larger than the variance of the prior. In our own experiments on MNIST we see that this improves the separation in predictive entropy. See the experiments section for details.

6 Evaluation

To measure the performance of the network we use the Intersection over Union (IoU), which is standard for segmentation tasks. The main goal is to compare the Bayesian implementation with the deterministic and see that the score is similar. Just like [1] and most works we don't evaluate voxels that are outside of the camera field of view. Unlike previous works we do evaluate on pixels outside of the room so that the network can learn to predict a room layout e.g. even though the room in just glimpsed through a door.

To evaluate if the Bayesian approach gives us more information about model uncertainty

Table 1: Compares Accuracy and mAP on MNIST for the deterministic and Bayesian mode of our simple network.

Mode	Accuracy	mAP
Bayesian, Cauchy(0.3), $\sigma_0 = 0.5$	97%	0.96
Deterministic, $\omega = 0$	**98%**	**0.97**
Deterministic, $\omega = 0.01$	97%	0.95

we look at the scores and uncertainty metrics for true positives (TP) and false negatives (FN) for each class or for different distances to the surfaces. If the knowledge about the model limitations is good, there should be a good separation in scores and/or uncertainty for TP and FN. We measure this with the Bhattacharyya coefficient (BC) [22], which is an approximation of the amount of overlap of two distributions. The data is split into N partitions, where in each partition we count the number of TP q_i and FN p_i and calculate the BC as

$$BC(\mathbf{p}, \mathbf{q}) = \frac{1}{N} \sum_{i=1}^{N} \sqrt{p_i q_i}.$$

Finally, we also compute the Mean Average Precision (mAP) as the area under the Precision-Recall curve. This measures both the separation and the accuracy of the model. We use 101 thresholds $T = \{0, 0.01, ..., 1.0\}$, just as the detection challenge COCO [23]. We follow their implementation and use interpolated precision to get a smoother curve.

7 MNIST experiments

For MNIST we created a simple 2D CNN using the same building blocks as we use in the 3D SSC-Net. We used the same code base and tools to allow for faster debugging and experiments. The network is trained in both Bayesian and Deterministic mode.

7.1 Baseline experiment

Here we have trained the network on the training set and evaluated on the test set. Table 1 shows the Accuracy and mAP for each mode. We see that the Deterministic CNN without weight decay, i.e. $\omega = 0$, performs best. Our Bayesian version is close behind with the benefit of having built in regularization in terms of a prior. By explicitly adding regularization in form of weight decay on the deterministic CNN we get the worst performance, especially noticeable in mAP.

7.2 Leave one out experiment

To see how well the uncertainty measurements captured model uncertainty we conducted an experiment where digit 0 was left out from the training data and then introduced at test time. Unless stated otherwise the Bayesian model uses Cauchy(0.3) as prior and $\sigma_0 = 0.5$ as initialization.

Prior and Initialization

In Figure 4 we see how the BC and mAP changes for different priors and σ_0 during training. A larger σ_0 decreases the BC score (increase separation) but require longer training times. By binning the results on both Entropy and mean score we get the best separation, which indicates that we can most likely train a classifier on these metrics and do better than the mean score. This is analogous with the GPOM [13] mentioned in section 2.

Activation function

Figure 5 shows how the choice of activation function affects the result. Just as [12] we observe that softplus in general yields better separation, both when used as layer activation and normalized as final activation.

Output distributions

For a more in depth understanding of the output we also look at the output distributions. As described in section 3.3 we have calculated the predictive mean $\bar{\mathbf{p}}_{ti}$ and entropy H_i for each sample indexed by i in the dataset. As usual we get the predicted label index as $l_i = \arg\max \bar{\mathbf{p}}_{ti}$ with corresponding score $\bar{\mathbf{p}}_{ti}(l_i)$ and entropy $H_i(l_i)$. In Figure 6 we form histograms of these for all samples in the test dataset categorized by their true class. For the deterministic case we see that the regularization helps separate the true zeros from the other predictions. However, the distributions for the other classes are much wider, indicating that the network is more uncertain overall. In the Bayesian case we see that the distribution for true zeros are well separated from the others, while maintaining certainty for the other classes. The entropy also seems to be a good indicator.

Table 2: Compares mIoU on SUNCG for the (D)eterministic and (B)ayesian versions of our CNNs.

Type	Arch.	Parameters	mIoU	mAP
B	SSC-Net	Cauchy(0.05), σ_0=0.05, β=0.1	0.22	0.26
B	UNet	Cauchy(0.1), σ_0=0.1, β=5	**0.23**	**0.29**
D	SSC-Net	ω=0	**0.23**	0.24
D	UNet	ω=0	0.21	0.21
D	SSC-Net	ω=0.01	0.15	0.19
D	UNet	ω=0.01	0.16	0.20

8 SUNCG mini experiments

SUNCG [1] is a large dataset consisting of 45,622 manually created synthetical indoor scenes with 84 labeled categories. We've used a subset of 2000 training scenes and 1000 testing scenes for the experiments. As in [1] we've used the flipped Truncated Signed Distance Function (TSDF) as input to our network and the same 11 categories as output. We chose the projective version since it is more realistic in a robotic setting. We've also chosen a coarser grid with 0.08m resolution and 60x40x60 in size which is similar to their output size and resolution.

8.1 Baseline experiment

In Table 2 we see the mean IoU (mIoU) over all categories for the Deterministic and Bayesian CNN. In general they have similar mIoU but the Bayesian versions have better mAP.

8.2 Complexity cost weight β

For the Bayesian UNet we conducted a β parameter experiment on SUNCG, which weighs the complexity cost against the likelihood cost. As a reference, the deterministic version with different weight decays are also included. We see in Figure 7 that $\beta = 5$ is better in all metrics but mIoU, where $\beta = 1$ has the highest score. We also see that all Bayesian versions are better than the deterministic in mAP and that a too large β makes the model unable to fit the data properly, while too small will get a better fit but worse separation.

Table 3: BC, mAP and mIoU for different network architectures when the *bed* class is removed from training. S=Score, E=Entropy. We observe that Bayesian SSC-Net has the best score in most metrics.

CNN	mIoU	mAP: S	mAP: E	BC: S	BC: E
SSC-Net ω=0	0.19	0.2		0.31	
SSC-Net ω=0.01	0.14	0.23		0.29	
UNet ω=0	0.2	0.14		0.39	
UNet ω=0.01	0.15	0.21		0.28	
B-SSC-Net	0.21	**0.26**	**0.19**	**0.27**	**0.28**
B-UNet	**0.21**	0.25	0.17	0.28	0.28

8.3 Leave one out

For the SUNCG mini dataset we removed all instances of the class *bed* during training and then tested on the full dataset. The result is presented in Table 3. Here we test both the SSC-Net and the UNet architecture in both Bayesian and Deterministic versions. We observe that both Bayesian versions outperform the Deterministic in mIoU, mAP and BC. For the sample based metrics the Bayesian SSC-Net outperform the Bayesian UNet, but this might not be the case given more data.

8.4 Output distributions

In Figure 8 we see histograms for true and false predictions at different distances from the observed surface for the Bayesian SSC-Net. We see that as the entropy (uncertainty) grows the ratio of false to true predictions increases, this is what we would expect from a sound measurement of uncertainty. It seems to have a better grasp of the uncertainty close to the observed surface.

8.5 Example output

See Figure 9 for example output. More examples are available in the supplementary material.

9 Conclusion

We've seen that a Bayesian CNN is clearly better at being uncertain when presented with unseen data. This was shown with a simple CNN model on the MNIST dataset. In the same experiment we also showed that the best separation between true and false predictions is found when combining entropy and mean score. This means that a final classifier layer could be added to get better probability estimates.

For the Semantic Scene Completion task we show that the Bayesian CNNs outperfom the Deterministic CNNs when faced with a category not seen at training time. This is shown by evaluating mIoU, mAP and separation score BC. Due to the complexity of the task and the model capacity we cannot show the same well separated output distributions as in the MNIST experiment. We do show that entropy represents the uncertainty well, especially close to the observed surface.

Future work will be to use an architecture better suited for sparse data and to utilize the RGB information from the image as well. We also want to look at how the uncertainty scores can be used in navigation.

Acknowledgment

This work is supported by Wallenberg AI Autonomous Systems and Software Program (WASP).

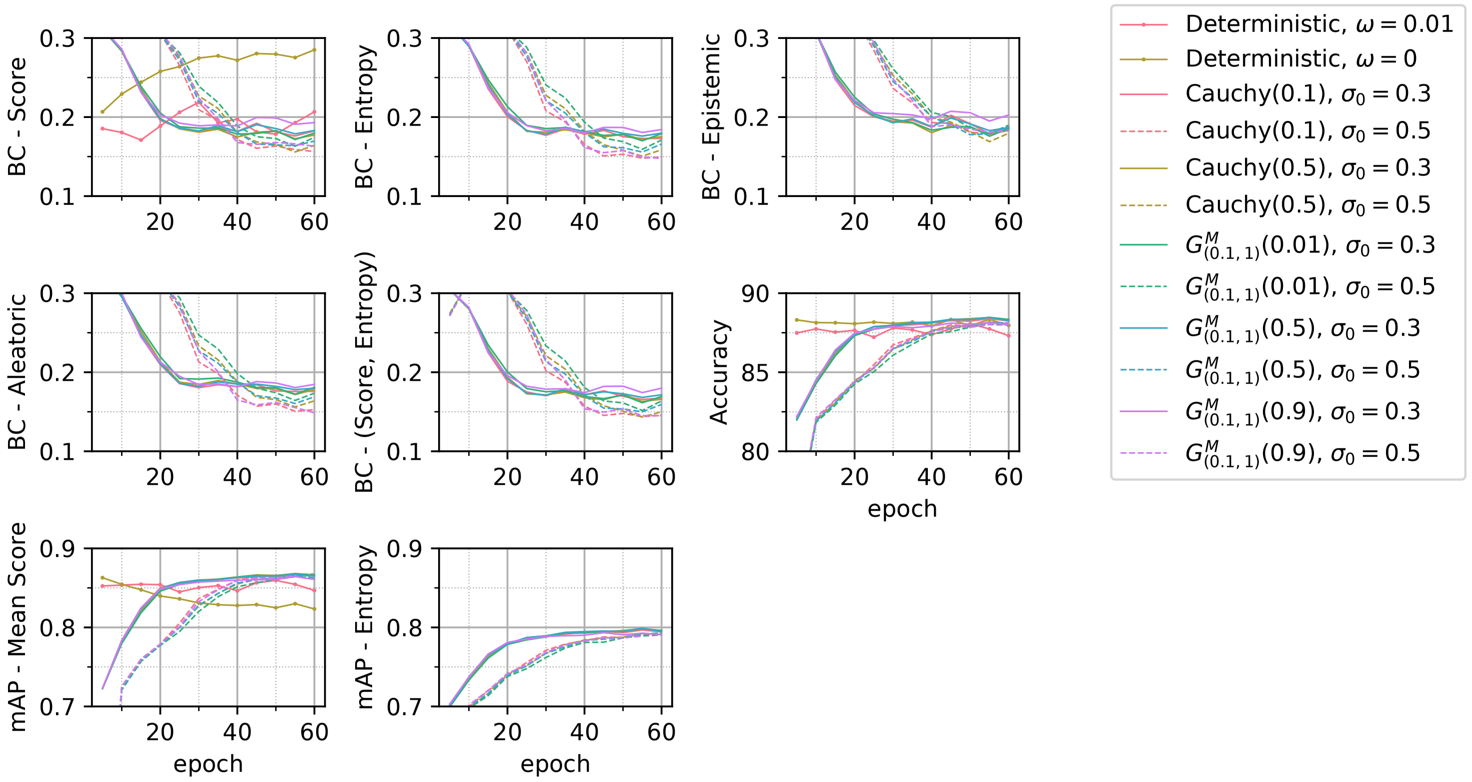


Figure 4: BC, mAP and accuracy for different configurations of prior and σ_0 for the MNIST experiment where digit 0 is introduced at test time. We see that high σ_0 seems to improve BC but increase training time and a prior which is more narrow than σ_0 gets a lower BC score. We also see that Entropy, Aleatoric uncertainty and Entropy coupled with score seems to have best separation. The Deterministic version are comparable in either mAP or BC, but not both at the same time.

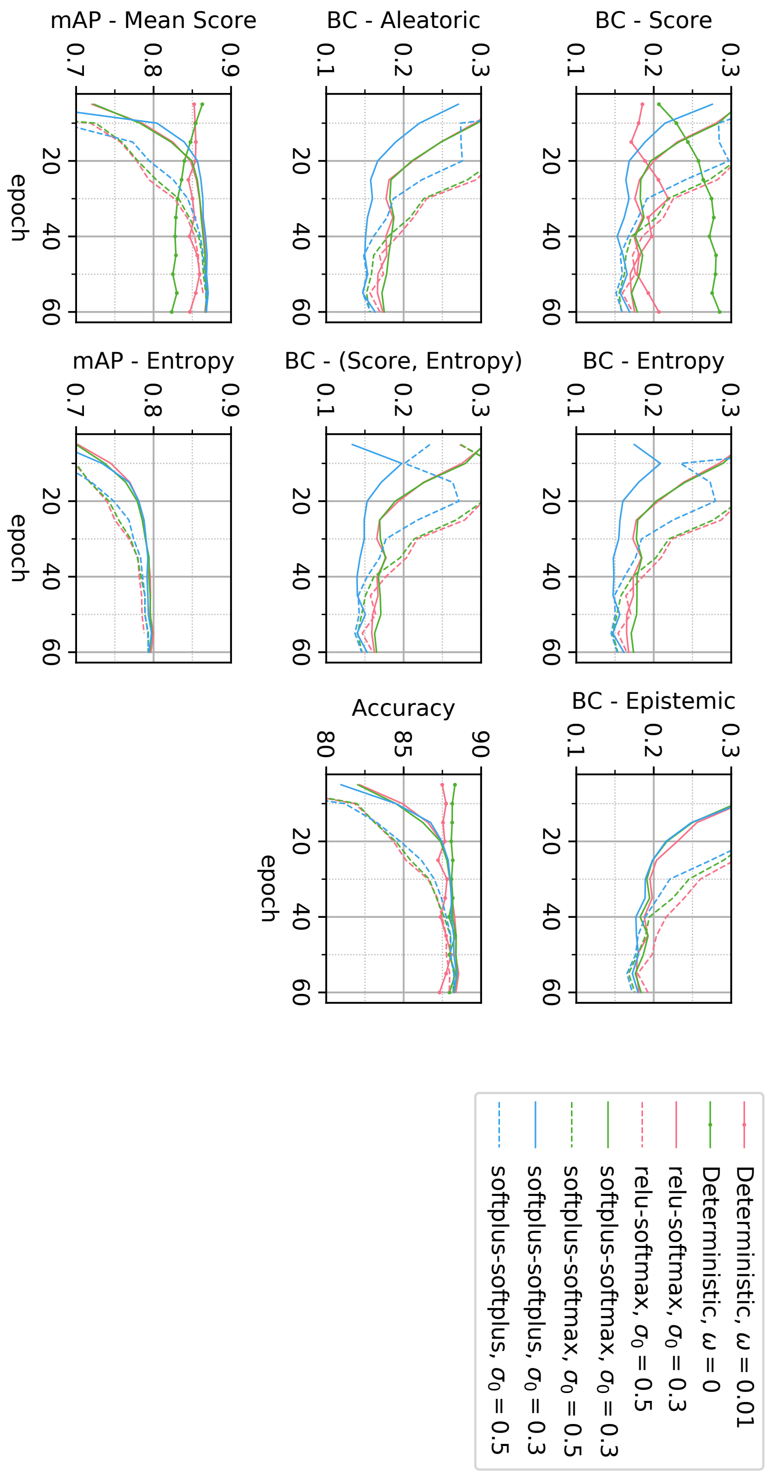
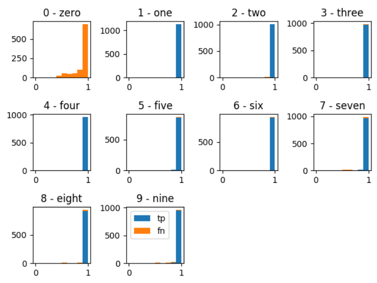
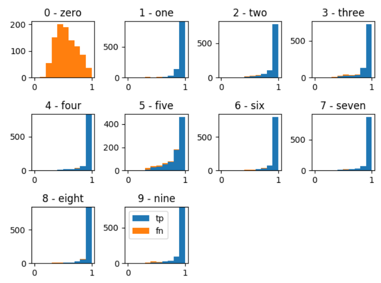


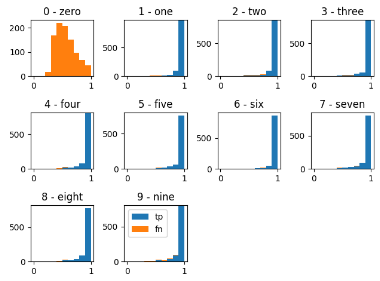
Figure 5: BC, mAP and accuracy for different activation functions and σ_0 for the MNIST experiment where digit 0 is introduced at test time. We observe that softplus as a layer activation function in general gets a lower BC score than ReLU. For the final activation the normalized softplus reaches a lower BC faster than softmax.



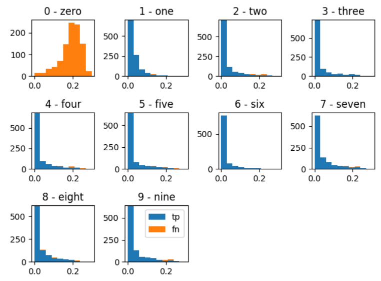
(a) Deterministic



(b) Deterministic Regularized



(c) Bayesian Mean Score



(d) Bayesian Entropy

Figure 6: Here we see true and false predictions for all digits in the test set when 0 has been left out from training. Blue bars are number of true predictions with the binned score, while orange are false predictions. In 6a we have the deterministic CNN which has high belief in many false 0 predictions. In 6b we have the deterministic CNN with regularization $\omega = 0.01$ which has a more balanced belief, but overall lower. In 6c and 6d we see lower certainty for digit 0 but still high certainty for other classes in both score and entropy respectively.

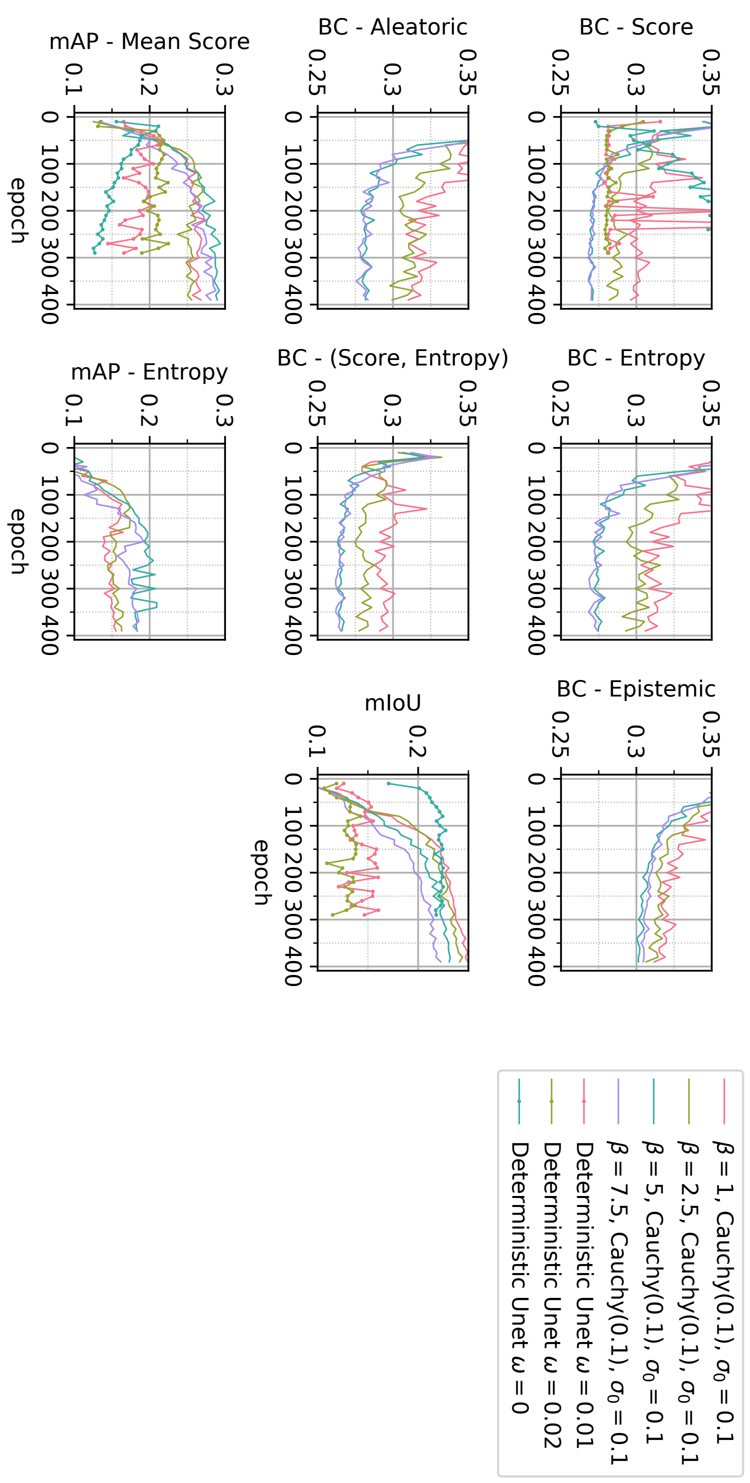
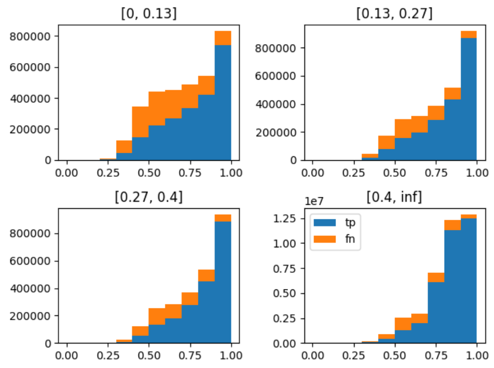
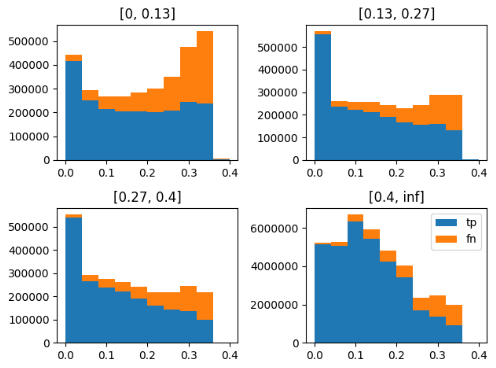


Figure 7: BC, mAP and mIoU for the Bayesian UNet with different weights β and ω for the SUNCG mini dataset. We observe that $\beta = 5$ is better in all metrics but mIoU, where $\beta = 1$ is best.



(a) Bayesian score



(b) Bayesian Entropy

Figure 8: Here we see true and false predictions for all voxels at different distances (in meter) from observed surfaces. In 8b we see that as the entropy (uncertainty) grows the ratio of false to true predictions increases. This hold for all areas, but especially close to the observed surfaces.

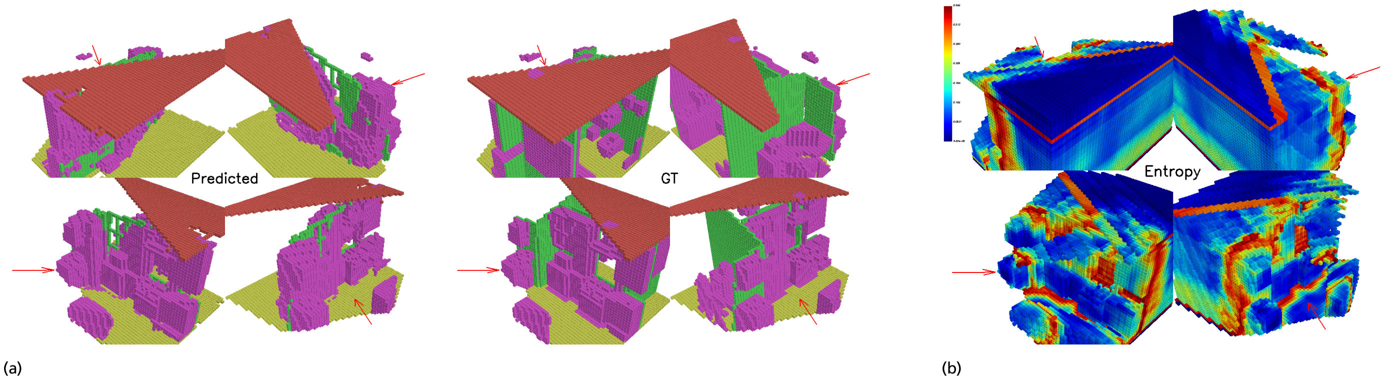


Figure 9: Here is one example output from the SUNCG mini test set. From the left we have predicted labels, ground truth and entropy.

References

[1] S. Song, F. Yu, A. Zeng, A. X. Chang, M. Savva, and T. Funkhouser, "Semantic scene completion from a single depth image," *Proceedings of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[2] M. Garbade, Y.-T. Chen, J. Sawatzky, and J. Gall, "Two stream 3d semantic scene completion," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2019.

[3] S. Liu, Y. HU, Y. Zeng, Q. Tang, B. Jin, Y. Han, and X. Li, "See and think: Disentangling semantic scene completion," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds. Curran Associates, Inc., 2018, pp. 263–274. [Online]. Available: http://papers.nips.cc/paper/7310-see-and-think-disentangling-semantic-scene-completion.pdf

[4] A. Dai, D. Ritchie, M. Bokeloh, S. Reed, J. Sturm, and M. Nießner, "Scancomplete: Large-scale scene completion and semantic segmentation for 3d scans," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] A. Dai and M. Nießner, "3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 452–468.

[6] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[7] B. Graham, M. Engelcke, and L. van der Maaten, "3d semantic segmentation with submanifold sparse convolutional networks," *CVPR*, 2018.

[8] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[9] W. Wu, Z. Qi, and L. Fuxin, "Pointconv: Deep convolutional networks on 3d point clouds," 2018.

[10] C. Choy, J. Gwak, and S. Savarese, "4d spatio-temporal convnets: Minkowski convolutional neural networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019, pp. 3070–3079.

[11] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural networks," *arXiv preprint arXiv:1505.05424*, 2015.

[12] K. Shridhar, F. Laumann, and M. Liwicki, "A comprehensive guide to bayesian convolutional neural network with variational inference," 2019.

[13] S. T. O'Callaghan and F. T. Ramos, "Gaussian process occupancy maps," *The International Journal of Robotics Research*, vol. 31, no. 1, pp. 42–62, 2012.

[14] V. Kuleshov, N. Fenner, and S. Ermon, "Accurate uncertainties for deep learning using calibrated regression," *arXiv preprint arXiv:1807.00263*, 2018.

[15] N. Seedat and C. Kanan, "Towards calibrated and scalable uncertainty representations for neural networks," 2019.

[16] M. Opper and C. Archambeau, "The variational gaussian approximation revisited," *Neural computation*, vol. 21, pp. 786–92, 10 2008.

[17] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" in *Advances in neural information processing systems*, 2017, pp. 5574–5584.

[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[19] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[20] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," 2015.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[22] A. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions," *Bull. Calcutta Math. Soc.*, vol. 35, pp. 99–109, 1943.

[23] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014.

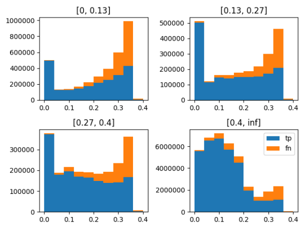
Appendix A. Output distributions SUNCG

In Figure 10 and 11 are histograms of output entropy and scores respectively for the 4 different network configurations in the SUNCG leave-one-out experiment in Section 8.3. There are 4 different histograms per network, each bins true (TP) and false positives (FP) with a certain distance from observed surface.

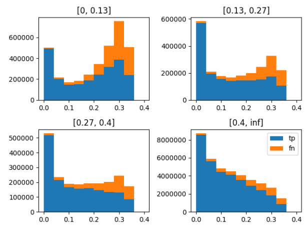
For the scores in 11 we see that in general the ratio of FP to TP grow as score decreases, at least close to the observed surface. For entropy the ratio grow as entropy increases, which we expect from a measurement of uncertainty.

Appendix B. Examples

In Figure 12 we show some example outputs from the Bayesian SSC-Net.

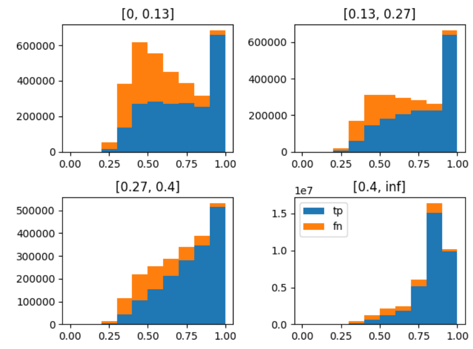


(a) Bayesian SSC-Net entropy

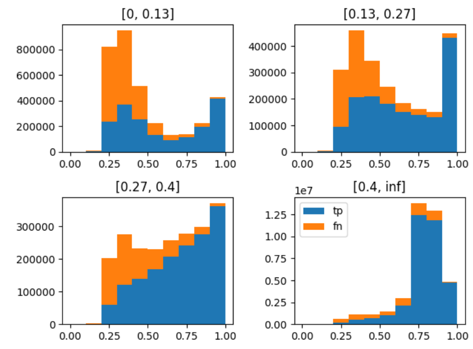


(b) Bayesian UNet entropy

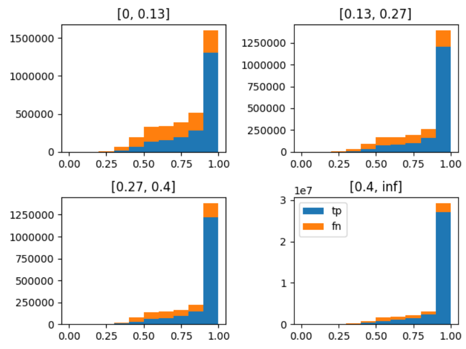
Figure 10: True and false predictions for all voxels at different distances (in meter) from observed surfaces. We see that as the entropy (uncertainty) grows the ratio of false to true predictions increases. This hold for all areas, but especially close to the observed surfaces.



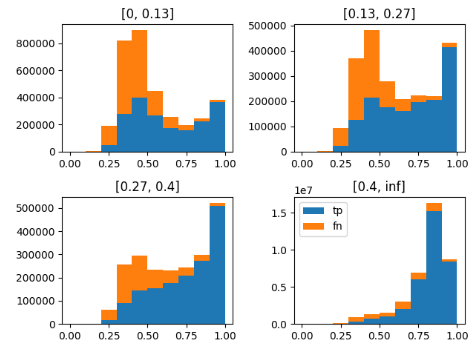
(a) SSC-Net score, $\omega = 0$



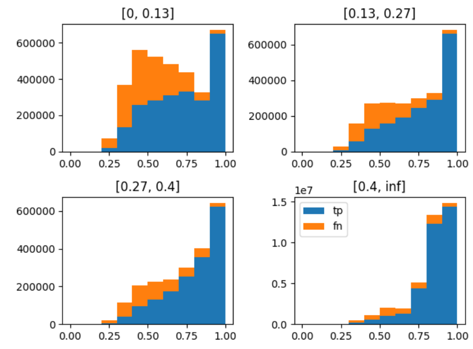
(b) SSC-Net score, $\omega = 0.01$



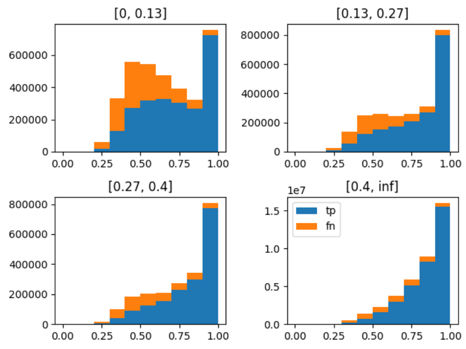
(c) UNet score, $\omega = 0$



(d) UNet score, $\omega = 0.01$

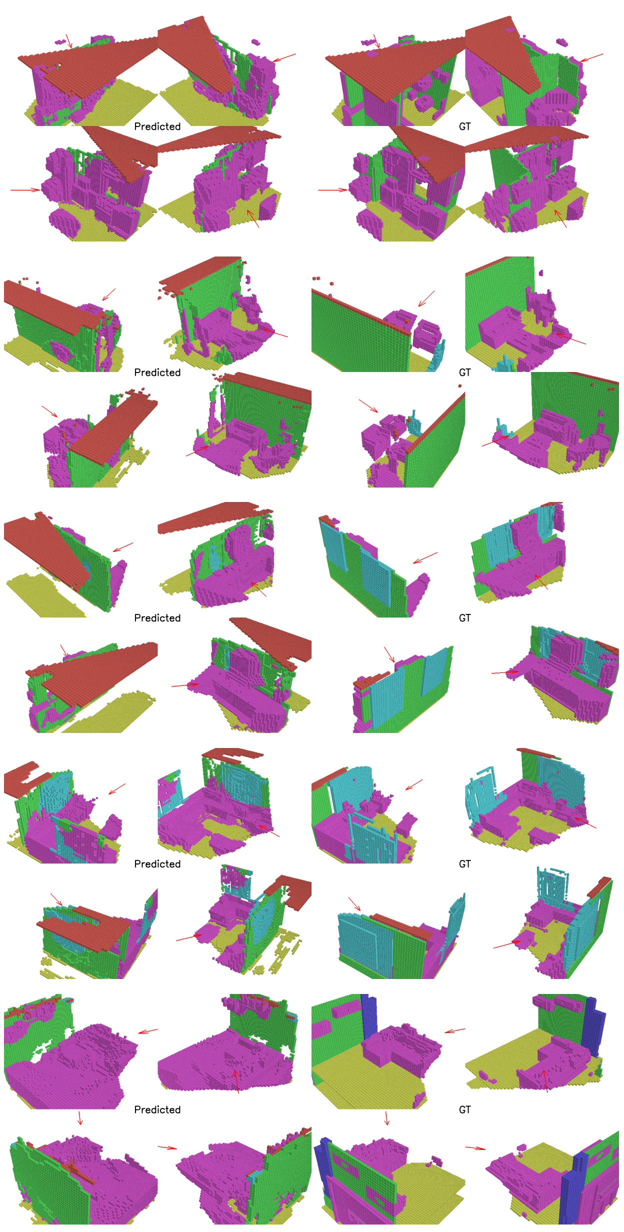


(e) Bayesian SSC-Net score

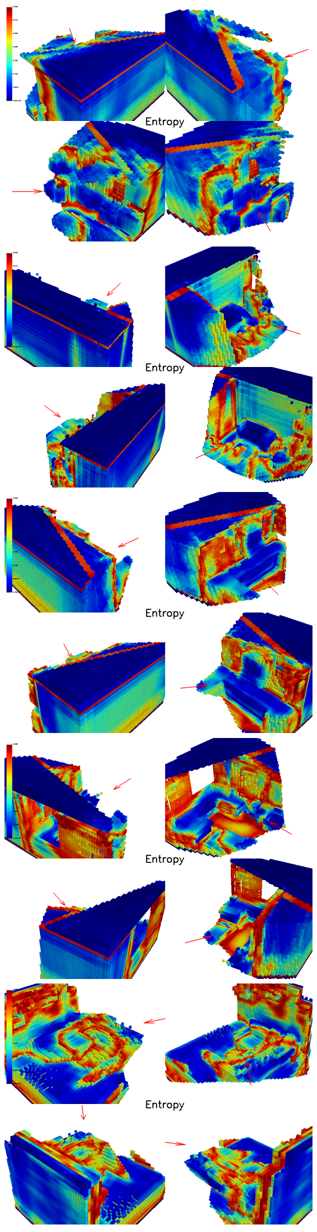


(f) Bayesian UNet score

Figure 11: True and false predictions for all voxels at different distances (in meter) from observed surfaces. We see the distribution of scores for all architectures.



(i)



(j)

Figure 12: Some example output from the Bayesian SSC-Net on the SUNCG dataset. From the left we have predicted labels, ground truth and entropy.

Paper II

Semantic Room Wireframe Detection from a Single View

David Gillsjö, Gabrielle Flood and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: Reconstruction of indoor surfaces with limited texture information or with repeated textures, a situation common in walls and ceilings, may be difficult with a monocular Structure from Motion system. We propose a Semantic Room Wireframe Detection task to predict a Semantic Wireframe from a single perspective image. Such predictions may be used with shape priors to estimate the Room Layout and aid reconstruction. To train and test the proposed algorithm we create a new set of annotations from the simulated Structured3D dataset. We show qualitatively that the SRW-Net handles complex room geometries better than previous Room Layout Estimation algorithms while quantitatively out-performing the baseline in non-semantic Wireframe Detection.

Keywords: Deep Learning, Room Layout Estimation, Line Segment Detection, Graph Convolution Networks

1 Introduction

Reconstruction of indoor spaces is useful for many applications, e.g. Virtual Reality, Real Estate sales and navigation for robots. Using consumer grade devices with a single camera (and possibly IMU) is preferred since this enables the application on a larger number of platforms. For this type of reconstruction one typically uses Structure from Motion (SfM) or Monocular Vision SLAM (Simultaneous Localization and Mapping). These rely on point features – points in the image with high saliency in terms of texture. This may be problematic in indoor spaces since most walls and ceilings are without salient textures.

This has been the inspiration for research on Room Layout Estimation, which is the task of estimating the Room Geometry in terms of planar surfaces. Often strong priors about the relation between the surfaces are used and Room Geometries are assumed to be too simple, see Figure 1. Furthermore the multi-view case is often handled using Panorama, which is not always applicable.

To relax the assumptions and handle multi-view scenarios we believe that the way forward is to develop several pipelines: semantic segmentation in the image, semantic segmentation

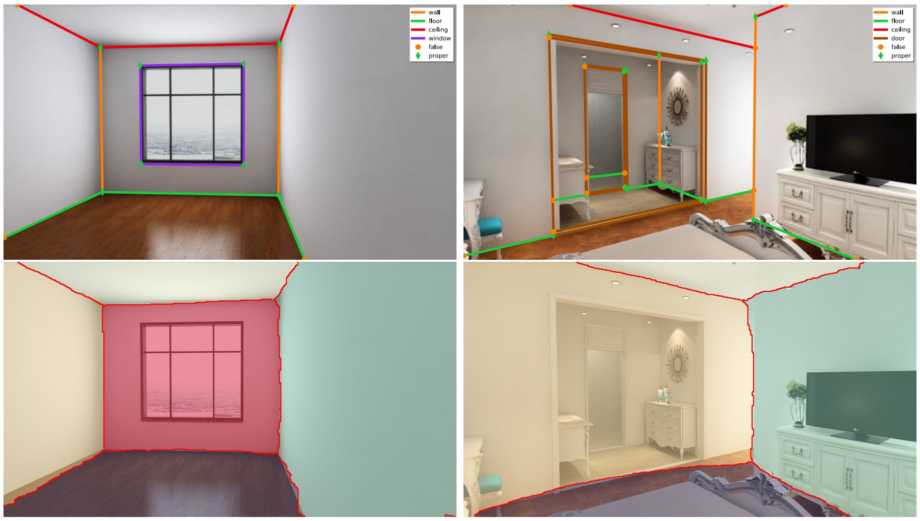


Figure 1: The images on top show the desired results from Semantic Room Wireframe detection and below are the results from a Room Layout estimation algorithm [1]. Lines are added between all detected planes for comparison. The algorithm produces satisfying results on simple scenes (left) but is lacking on more complex scenes, e.g. to the right where the wide opening is erroneously classified as part of the wall.

in 3D, image object detection, corner detection and semantic line detection. These could then be used together – ideally in an end-to-end fashion. In this work our aim is to concentrate on one of these aspects and propose a Semantic Room Wireframe Detection task, where connected semantic line segments forming a wireframe, are to be detected. Since this is a detection task no strong assumptions are made and the detections can be used in a larger optimization framework to handle more complex Room Geometries captured from multiple views. Compared to previous methods we for example include labels for neighbouring rooms visible through open doors and label doors and windows in addition to the room structure.

We generate annotations, which consist of semantic line segments and semantic junctions from the synthetic Structured3D dataset. To perform the detection we adapt HAWP [2], which is a CNN (Convolutional Neural Network) developed for the Wireframe Detection task, and extend it with a Graph Convolutional Network (GCN) [3] module. We call the new network SRW-Net (Semantic Room Wireframe Network). We show qualitatively that the SRW-Net[1] handles complex room geometries better than previous Room Layout Estimation algorithms while quantitatively out-performing HAWP [2] and L-CNN [4] for non-semantic Wireframe Detection. **The contributions of this paper are:**

- Proposal of the Semantic Room Wireframe detection task.
- New annotations for this task which we call Structured3D-SRW.

[1]See `https://github.com/DavidGillsjo/SRW-Net` for code and data.

- Implementation and evaluation of a Neural Net for Semantic Room Wireframe detection, SRW-Net.

2 Related Work

Room Layout Estimation was studied early by Hedau *et al.*[5] which used the Manhattan World assumption [6] as prior for the room shape. Early methods [7–9] used a pipeline which first extracted handcrafted features, then did vanishing point detection and lastly hypothesis generation and ranking. Recent approaches use CNNs, e.g. Mallya and Lazebnik [10] use structured edge detection forests, and a CNN to predict edge probability masks. Lin *et al.*[1] propose an end-to-end CNN with adaptive edge penalty and smoothness terms for pixelwise segmentation of the room image. RoomNet [11] directly predicts ordered keypoints in a room layout and DeepRoom3D [12] use an end-to-end CNN to predict a cuboid. Most recent methods [13–16] use CNNs to predict edges and then optimize for the Room Layout using geometric priors. The datasets mostly used are LSUN [17] and Hedau [5].

Zhang *et al.*[18] advocated the use of Panorama images for increased performance in Room Layout Estimation. This has since been an inspiration for Panoramic Room Layout Estimation, which allows for better use of context and prior knowledge about camera orientation and calibration. Algorithms are typically based on CNNs [19–21] using different image representations and post-processing. For example CFL [22] predicts the 3D layout from a spherical image using edge and corner maps. Despite the promising results from these methods Room Layout Estimation from panorama does, indeed, limit the use of more conventional images. Common datasets for panorama are SUN360 [23], Stanford (2D-3D-S) [24] and Structured3D [25].

Recently, general Room Layout Estimation from a single perspective image, which does not assume cuboid shape, has been studied. [26] solves a discrete optimization problem over 3D polygons using both RGB and Depth information. [27] use plane detection to form a 3D model over a video sequence. [28] use a combination of plane, depth and vertical line detections to estimate a general Room Layout. Neither of these handle detections through open doorways. Data for this task is Structured3D [25] and ScanNet [29].

There is a vast literature on the use of lines for 3D understanding, even though SfM and SLAM are most often used in terms of points. For example understanding consistency constraints [30], exploiting such constraints for 3D understanding and SfM methods using only lines [31–37].

Detection of connected line segments has been studied as the task of Wireframe Estimation [2, 38–40]. However, these methods do not try to use semantic understanding of the lines and junctions and do not exploit such semantic understanding in terms of Room Layout Estimation. While recent works mostly form the Wireframe using junction proposals some works use line predictions directly. For example in [41], a combination of line predictions and graph networks are used. YorkUrban [42] and ShanghaiTech [43] are common datasets. Other approaches that predict semantic information for line segments [44] are focused around object detection rather than detection of indoor room structures.

3 Semantic Room Wireframe Detection

We define a Semantic Room Wireframe Detection task. The prediction is done from a single perspective image and should result in a number of semantic lines in the image which – when connected by junctions – form a wireframe marking the intersections between the planes defining the room layout.

The task can be seen as a combination of Room Layout Estimation and Wireframe Detection. It is more flexible than Room Layout Estimation since no priors on the room structure are used but also contains more information than regular Wireframe Detection. The results can, e.g., be used to aid reconstructions, s.a. SfM and SLAM where salient textures are missing. It is also suitable for multi-view Room Layout Estimation, where it gives more flexibility for global optimization and better information in door openings. For an example of the desired output of a Semantic Room Wireframe Detector, see the top row of Figure 1.

4 Structured3D-SRW

Our annotations for this task, which we call Structured3D-SRW, are generated from Structured3D [25] which is a large-scale, photo-realistic simulated dataset with 3D structure annotations. It consists of 3500 scenes, with a total of 21’835 rooms and 196’515 frames. Structured3D contains 3D information about the room structure and is built up from junctions, lines and plane polygons. A junction is a point in $\mathbb{R}^3$ and a line is defined as the line segment formed by two junctions. A plane polygon is defined by a set of connected line segments and its parameters π defining the normal vector and plane equation. These planes are labeled with semantic information, for example if the plane is a floor and part of a kitchen.

There are both panoramic and perspective images in the dataset. For each room, there are configurations both without furniture, with a little and with a lot of furniture. For each of the images, there is also a depth map and a semantic mask. The data is split so there are 3000 training, 250 validation and 250 testing scenes.

4.1 Data Analysis

Our annotations are generated by combining the 3D information, semantic masks and perspective RGB images with full furnishing. For the images with full furnishing there are 202 scenes missing from the training set, so that leaves us with 2'798 training scenes. While working with the data, we identified three main issues.

Firstly, there are a few planes to which only two junctions belong. However, only two junctions cannot uniquely define a plane. Secondly, it turns out that there are a number of planes for which the defining junctions do not lie in the plane. We calculated the Euclidean distance between all plane junctions and the plane according to supplied parameters and the result can be seen in Figure 2. We see that there are around 10^3 planes with a maximum distance larger than 5mm. This is problematic since the assumption is that all surfaces are planar in this synthetic dataset. Thirdly, there is no information about whether the doors in Structured3D are open or closed. Since one of the strengths of our proposed method is that more complex structures can be captured, we want the method to identify open doors to create a wireframe also in the room behind them. One example of this can be seen in Figure 3, where the large opening in the left wall is annotated as a door. Knowing that it is open, we can train our method to detect the structure of the second room as well, as annotated in the image.

We have implemented the following measures to adjust for these three main issues:

(i) For the walls with only two junctions, we cannot form plane polygons to make occlusion checks. Because of this we simply discard the scenes containing such planes.

(ii) To verify that junctions lie in the plane, we chose to estimate new plane parameters from the junctions by solving the DLT problem

$$\min_{\|\pi\|^2=1} \|M\pi\|^2, \text{ for } \quad M = \begin{pmatrix} p_1 & p_2 & \dots & p_n \end{pmatrix}^T, \tag{1}$$

$$p_i = \begin{pmatrix} x_i & y_i & z_i & 1 \end{pmatrix}^T, \tag{2}$$

where p_i is a junction coordinate and π are the four plane parameters s.t. a point q is in the plane iff. $q^T\pi = 0$. With these plane parameters we get lower maximum and median distances, compared to using the supplied parameters. Still, there are many planes that have large distances to their junctions. We therefore remove any scene with planes that have maximum error larger than 1mm. **(i)** and **(ii)** combined leaves us with 2'502, 218 and 205 scenes corresponding to 57'252, 5'684 and 5'085 images for training, validation and testing, respectively.

(iii) We do the following to determine whether doors are open or closed: For each door polygon $D_{i,j}$ belonging to wall plane π_i we take a set of 100 uniformly distributed sample

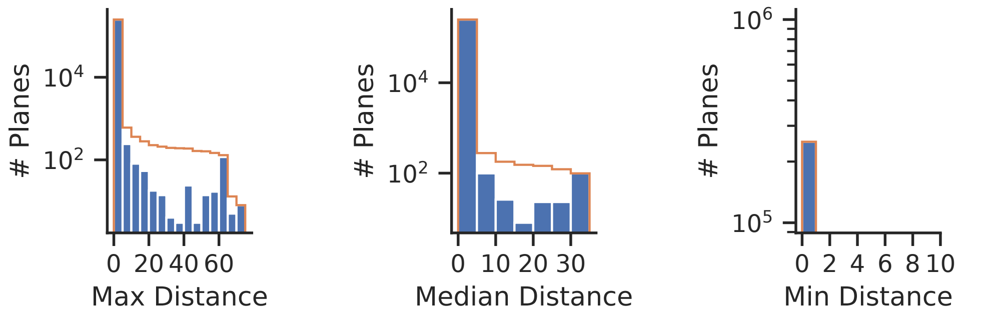


Figure 2: Histograms for the maximum, median and minimum distance from each plane to its junctions using the supplied plane equation. Note that the plane parameters are not optimized to minimize distance to all junctions.

points $\hat{D}_{i,j} = \{p_k \in D_{i,j}\}$. For each sample set $\hat{D}_{i,j}$ and each perspective image $\hat{I}_l$ for a scene, we find the sample points

$$\hat{D}^l_{i,j} = \left\{ (x_k, y_k, 1)^T \sim P_l p_k \;\middle|\; (x_k, y_k) \in \hat{I}_l \right\}, \tag{3}$$

in pixel coordinates using the camera matrix $P_l = K_l[R_l \; t_l]$, where K_l are intrinsic parameters, R_l rotation matrix and t_l translation vector. We then compute a door closed ratio

$$c_{i,j} = \frac{\sum_l \sum_k \alpha_{lk}}{\sum_l \sum_k 1}, \qquad \alpha_{lk} = \begin{cases} 1, & \text{if } \hat{I}_l[y_k, x_k] = \textit{door}, \\ 0, & \text{otherwise}, \end{cases} \tag{4}$$

and say that the door $\hat{D}_{i,j}$ is closed if $c_{i,j} > 0.3$.

4.2 Semantic Information

In Structured3D each plane is annotated with a type and a semantic class. By combining the information in these two fields we mark each plane as either *wall*, *floor*, *ceiling*, *door* or *window*.

We choose to have one label per line and define each line label as the combination of the two planes it belongs to. If we look at the last column of Table 1 we see that there are seven occurring classes, where *door* and *window* planes yield two different line labels each. Since most of the *door-door* and *window-window* lines will be concealed behind doors and window glass we choose to merge these with *door-wall* and *window-wall*, respectively. We then get the mapping in Table 1, which consists of the same five labels as for the planes, namely *wall*, *floor*, *ceiling*, *door* and *window*.

Furthermore, we also label the junctions. The dataset consists of two different kinds of junctions. There are junctions corresponding to a 3D junction where three planes meet;

Table 1: Mapping of line labels from plane labels.

Line label	Plane labels	Occurrence [%]
door	*door-wall*	20.4
wall	*wall-wall*	15.5
ceiling	*ceiling-wall*	15.5
floor	*floor-wall*	15.5
door	*door-door*	11.6
window	*window-wall*	11.0
window	*window-window*	10.4

these we call *proper* and they should be detected even when occluded by furniture. There are also junctions which occur either due to the camera having limited field of view, or due to occlusion from other planes. These we call *false*. See Figure 3 for an example.

5 SRW-Net

For the CNN architecture we chose to base it on HAWP (Holistically-Attracted Wireframe Parsing), presented in Xue *et al.*[2]. Since our data contains semantic labels for both lines and junctions, but fewer lines per image, we had to make some adjustments to the architecture and parameters. In addition to the changes we introduce a GCN module, trained separately, to refine the features using neighbours in the wireframe. See Figure 4 for an overview of the architecture.

5.1 Backbone

Before fed into the network, the image is resized to $H \times W$ pixels, where $H = W = 512$. It is then passed through a backbone Hourglass Network [45] which generates a latent feature space $\mathcal{F}$ of size $N \times \hat{H} \times \hat{W}$, where $\hat{H} = \hat{W} = 128$ and $N = 128$ is the number of channels. These features are then used by the different network heads.

5.2 Junction Proposals

The junction proposal head predicts from $\mathcal{F}$ a junction label matrix $\mathcal{J}$ of size $3 \times \hat{H} \times \hat{W}$ with one-hot encoding for labels *invalid*, *false*, *proper*. It also predicts an offset matrix $\mathcal{O}$ of size $2 \times \hat{H} \times \hat{W}$. They are computed using a 1×1 convolutional layer and then sigmoid for $\mathcal{O}$ and softmax for $\mathcal{J}$. This is different from HAWP since they did not have labels for

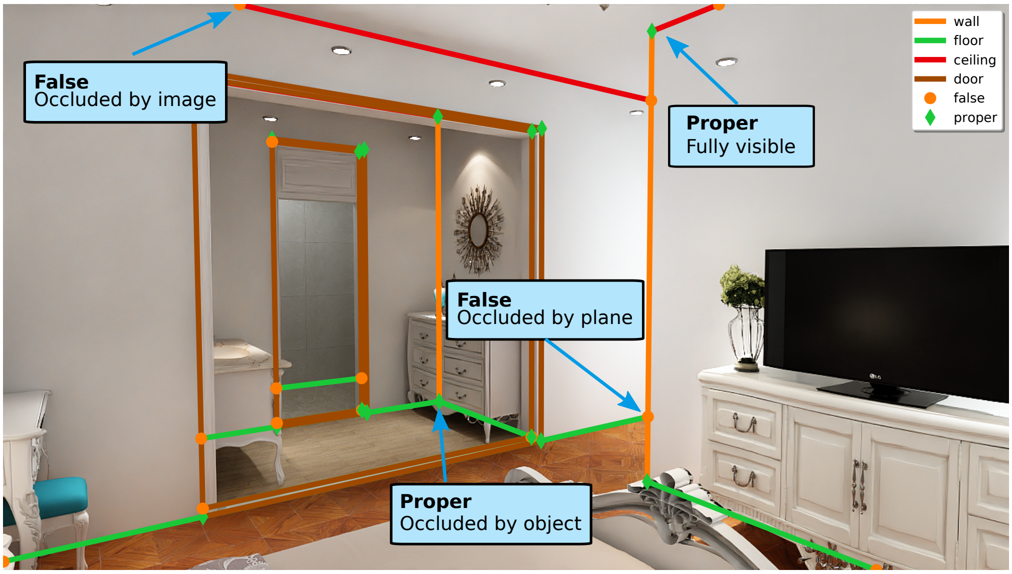


Figure 3: Example cases for junction labels *proper* and *false* .

junctions and used sigmoid for $\mathcal{J}$. We then form $\mathcal{J}_v = 1 - \mathcal{J}[invalid]$. After a 3×3 NMS (Non-Maximum Suppression) to $\mathcal{J}_v$ the top K bins $\mathcal{J}_K = \{(x_k, y_k), k = 1...K\}$ are selected ($K = 80$) and the offset map is used to compute the final position of the junctions $\mathcal{J}_f = \{(x_k, y_k) + \mathcal{O}(x_k, y_k) \cdot w, k = 1...K\}$ where w is a rescaling factor. Like HAWP we use cross entropy loss to form $\mathbb{L}(\mathcal{J}, \hat{\mathcal{J}})$ and L1 loss for $\mathbb{L}(\mathcal{O}, \hat{\mathcal{O}})$. But we weight *invalid* as 1:250 in $\mathbb{L}(\mathcal{J}, \hat{\mathcal{J}})$ to account for bias and improve recall. The final loss function is $\mathbb{L}_j = \lambda_{msk} \cdot \mathbb{L}(\mathcal{J}, \hat{\mathcal{J}}) + \lambda_{off} \cdot \mathcal{J}_v \odot \mathbb{L}(\mathcal{O}, \hat{\mathcal{O}})$, where $\odot$ is elementwise multiplication, λ_{msk} and λ_{off} are design parameters.

5.3 Line Segment Proposal

The line segment proposal head is the same as for HAWP and is a modification of AFM [39] which uses a parametrization of line segments using four parameters. It is, however, not unique and each parametrization may yield up to three line segments. The proposal head takes $\mathcal{F}$ and predicts a $4 \times \hat{H} \times \hat{W}$ matrix, where the first dimension is the 4D AFM parametrization, which is then converted to the endpoint line segment representation. The loss function is the same as for HAWP and we denote it $\mathbb{L}_{LS}$.

5.4 Line Segment and Junction Proposal Matching

Now we have line segment proposals and junction proposals. Each line segment's endpoints are matched to their closest junction. If the Euclidean distance between each endpoint and matched junction is below $\tau = 10$, we form a line segment between these two junctions. We denote the set of matched line segments $\mathcal{L}_m$.

5.5 Matched line segment classification

Each matched line segment $l \in \mathcal{L}_m$ will be classified as either *invalid*, *wall*, *floor*, *ceiling*, *window* or *door*. Following the implementation of HAWP we use the LoIPool operation [4] from L-CNN. First $s = 32$ uniformly spaced points along the line are sampled from $\mathcal{F}$. All features are then reduced, concatenated and given to the 2-layer fully connected classifier ending with the softmax operation and cross entropy loss $\mathbb{L}_{cls}$. This is trained end to end and the final loss is $\mathbb{L} = \mathbb{L}_{junc} + \mathbb{L}_{LS} + \mathbb{L}_{cls}$. During training we balance negative and positive examples while also providing hard negative examples from the ground truth data, see HAWP [2] for details. We reduced the number of sampled lines to 100 negative and 100 positive.

5.6 Wireframe Refinement - Graph Convolutional Network

This GCN module uses the graph structure to augment the line and junction features with encoded features from the neighbouring lines and junctions prior to classification. The module is trained separately from the larger network and takes lines with *invalid* score less than 0.95 as input. We form two graphs, the junction graph is simply the estimated wireframe where junction features are nodes and lines define edges. The line segment graph is created by taking the Line Graph [46] of the junction graph, s.t. the line segment features now are nodes. The architecture of the GCN layers are encoder-based with skip-connections to a fully connected classifier layer, see Figure 4. We use 4 GCN layers for junction and line head respectively. To achieve good results we sample positive and negative examples as 5:1 for the loss function. The loss and classifier architecture are the same as previously.

6 Evaluation

sAP (structural average precision) [4] is the area under the precision and recall curve for a set of scored detected line segments. Let $\hat{L} = \begin{pmatrix} \hat{p}_1 & \hat{p}_2 \end{pmatrix} \in \mathcal{L}_g$ be any line segment in the set of true line segments. A detected line segment $L = \begin{pmatrix} p_1 & p_2 \end{pmatrix}$ is a true positive if

$$\min_{\hat{L} \in \mathcal{L}_g} \delta(L, \hat{L}) = \|p_1 - \hat{p}_1\|_2^2 + \|p_2 - \hat{p}_2\|_2^2 \leq \beta, \tag{5}$$

where β is a design parameter. We also require that only one detection is matched to each true line segment. Any extra predictions are marked as false positives. In our experiments we evaluate the metric at $\beta = 5, 10, 15$ at 128×128 resolution and denote the metrics sAP^5, sAP^{10} and sAP^{15} respectively. We calculate sAP for each separate label and then take the mean to form msAP. We also calculate $\text{sAP}^m = \sum_\beta \text{sAP}^\beta / 3$ for each label and take its mean across labels as msAP^m.

NMS (Non Maximum Suppression) is also performed on line segments to improve performance. We use the same distance δ as in Equation (5) and say that if a line segment L has a neighbour $\hat{L}$ s.t. $\delta(L, \hat{L}) < \gamma^2$ with the same predicted label and higher score, L is removed. We use $\gamma = 3$.

jAP (junction AP), is analogous to sAP. Instead of the criteria in Equation (5) we take the Euclidean distance between the junction and the closest ground truth junction of the same label. The thresholds are 0.5, 1.0 and 2.0 and mjAP^m, jAP^m follows as before.

7 Experiments

All experiments were done using pre-trained weights from HAWP as initialization for applicable layers, as we noticed this produced better results than training from scratch. Models were trained for 40 epochs with learning rate $4 \cdot 10^{-4}$, weight decay 10^{-4} and batch size 11 on a Nvidia Titan V. The learning rate was reduced to $4 \cdot 10^{-5}$ at epoch 25. For the GCN refinement module we trained for 10 epochs with learning rate $1.2 \cdot 10^{-3}$, weight decay 10^{-4} and batch size 60. The learning rate was reduced to $1.2 \cdot 10^{-4}$ at epoch 5.

7.1 Model evaluation

We evaluate the proposed model on the test set. See Table 2 for AP numbers and Figure 5 for the PR (precision recall) curve for threshold 10. We see that *floor* is by far the most difficult line type to detect correctly. Since there is no method available for direct comparison we trained HAWP [2] and L-CNN [4] on Structured3D-SRW using their code and training methods. All models are initiated with the pre-trained weights provided by the code releases. Since these algorithms output a non-semantic wireframe we generate a wireframe prediction from our method by calculating the score as $(1 - \textit{invalid})$ for junctions and lines. In Table 3 we see that our method out-performs both HAWP and L-CNN in line segment detection (sAP). HAWP performs better in the junction metric jAP.

Table 2: AP Scores for the final model.

Type	sAP^5	sAP^{10}	sAP^{15}	sAP^m
ceiling	35.5	42.4	46.5	41.5
door	44.7	49.6	52	48.8
floor	15.5	21.6	24.6	20.6
wall	39.8	47.3	51.2	46.1
window	49.5	55.4	58.2	54.4
mAP	37	43.3	46.5	42.3

Type	$jAP^{0.5}$	jAP^1	jAP^2	jAP^m
false	17.8	36	43.7	32.5
proper	21.9	39	46.8	35.9
mAP	19.9	37.5	45.3	34.2

7.2 Ablation study

In Table 4 we show results on how the class weights for the junction loss $\mathbb{L}(\mathcal{J}, \hat{\mathcal{J}})$, the GCN refinement module and line NMS affects the classification for the validation set. Although equal weights for $\mathbb{L}(\mathcal{J}, \hat{\mathcal{J}})$ outperforms a weighted loss in sAP^m the GCN refinement module benefits from the improved jAP^m and recall. Therefore the weighted loss with GCN refinement module and line NMS has the best performance.

7.3 Room Layout vs Semantic Room Wireframe

We make a qualitative comparison in Figure 6 with the method of Lin *et al.*[1][2] from the Room Layout Estimation task to illustrate the benefits of this data. For our method we use all lines with score higher than 0.9. Since the methods are trained on different data we cannot compare performance, but we see that the representation for the Room Layout Estimation task is not sufficient for these complex scenes.

8 Conclusions

We have introduced the task of Semantic Room Wireframe detection and generated annotations from Structured3D. This dataset challenges algorithms to detect complex Room Geometries from single view perspective images. We show that a CNN for Wireframe estimation can be adapted to this task and handles complex Room Geometries better than a reference method for Room Layout Estimation. We also show how the graph structure can

[2] https://github.com/leVirve/lsun-room

Table 3: Comparision of the (1-*invalid*) score against HAWP and L-CNN.

Arch.	sAP^m	jAP^m
LCNN	31.8	34.9
HAWP	46.0	**37.0**
SRW	**46.6**	36.2

be used with a GCN refinement module to out-perform the baseline even in non-semantic wireframe detection.

Acknowledgment

This work was partially supported by the strategic research projects ELLIIT and eSSENCE, the Swedish Foundation for Strategic Research project, Semantic Mapping and Visual Navigation for Smart Robots (grant no. RIT15-0038) and Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

Table 4: Ablation study on validation set.

GCN	NMS	$\mathbb{L}(\mathcal{J}, \hat{\mathcal{J}})$	sAP^m	jAP^m
		equal	44.6	26.4
		weighted	31.6	37.4
✓		equal	45.0	**39.0**
✓		weighted	45.3	37.9
✓	✓	equal	45.8	37.9
✓	✓	weighted	**46.7**	37.9

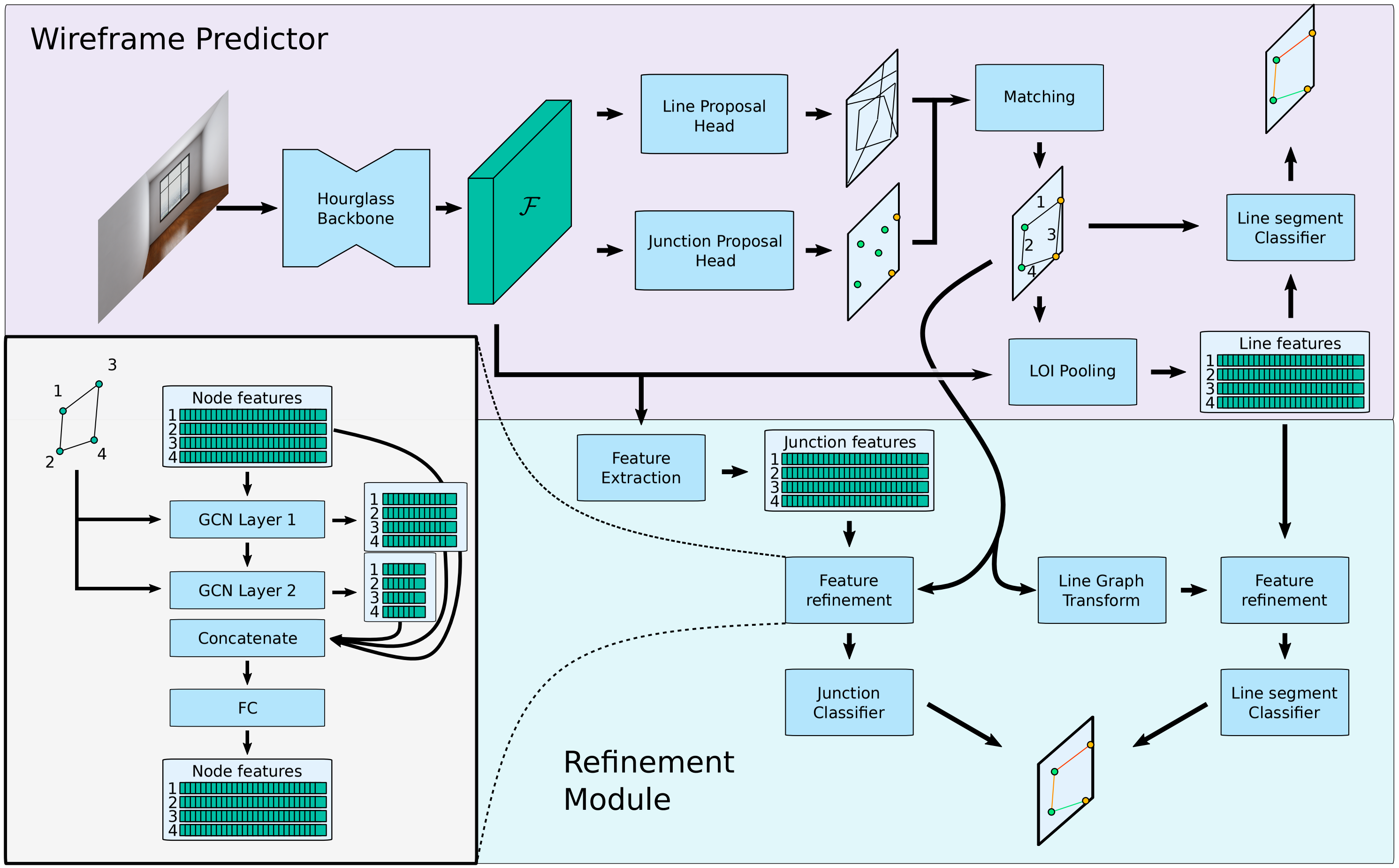


Figure 4: An overview of the SRW-Net architecture.

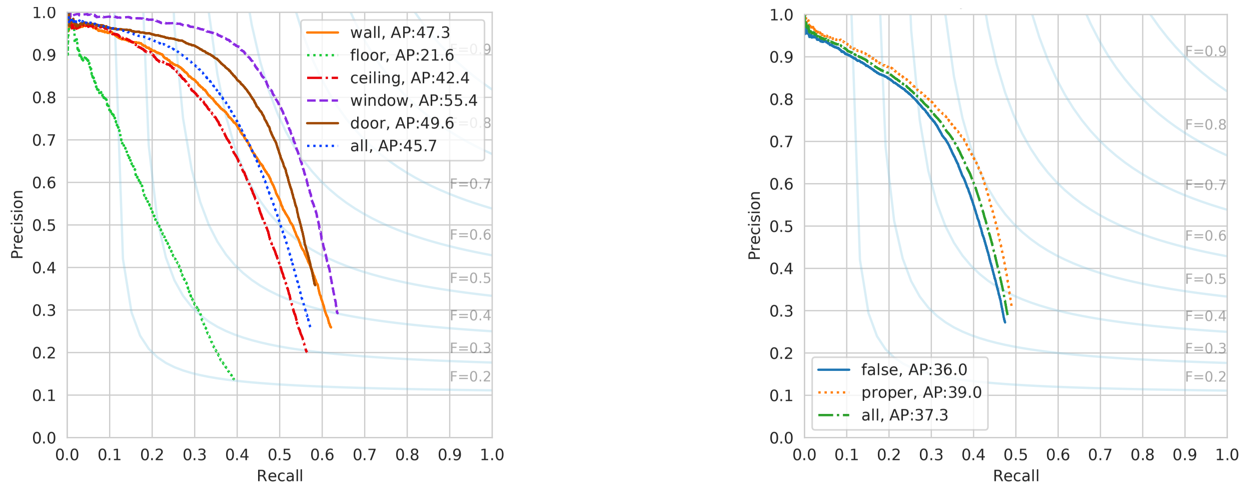


Figure 5: Left: sAP^{10} PR curve for line segments. **Right:** jAP^{1} PR curve for junctions. Note that *floor* is the most difficult line label to predict. While all junction labels are equally difficult.

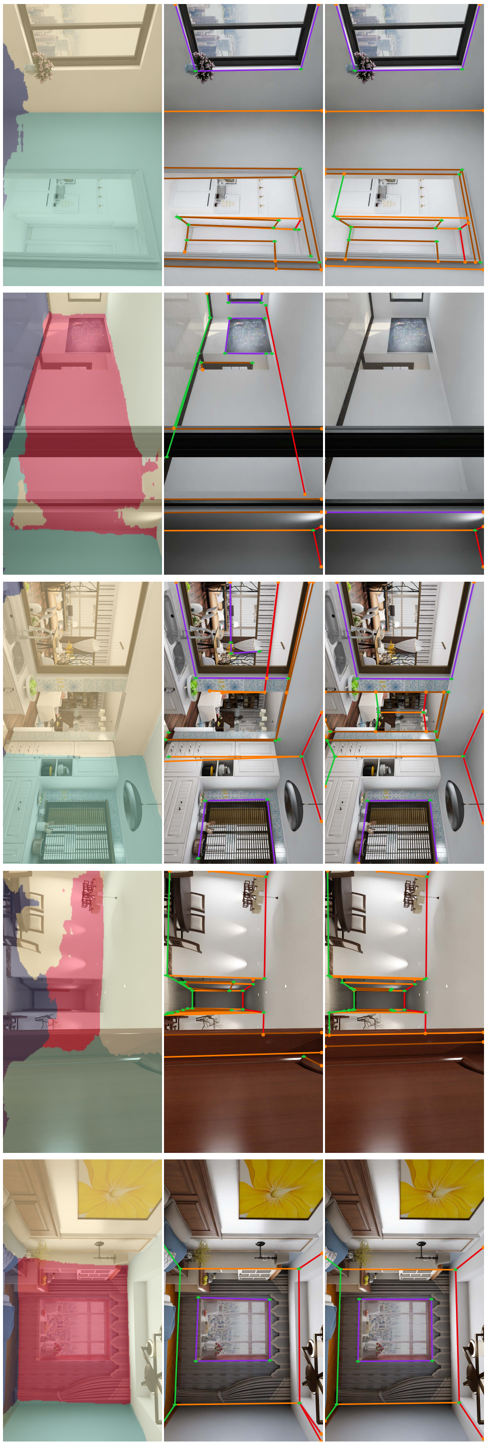


Figure 6: Results from the comparison with Room Layout Estimation. On top: the original images with Semantic Wireframe annotations. In the middle: the results from our method. In the bottom: Room Layout results from [1].

References

[1] H. J. Lin, S.-W. Huang, S.-H. Lai, and C.-K. Chiang, "Indoor scene layout estimation from a single image," in *2018 24th International Conference on Pattern Recognition (ICPR)*, 2018.

[2] N. Xue, T. Wu, S. Bai, F.-D. Wang, G.-S. Xia, L. Zhang, and P. H. Torr, "Holistically-attracted wireframe parsing," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[3] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*, 2017.

[4] Y. Zhou, H. Qi, and Y. Ma, "End-to-end wireframe parsing," in *ICCV 2019*, 2019.

[5] V. Hedau, D. Hoiem, and D. Forsyth, "Recovering the spatial layout of cluttered rooms," in *2009 IEEE 12th international conference on computer vision*. IEEE, 2009, pp. 1849–1856.

[6] J. M. Coughlan and A. L. Yuille, "Manhattan world: compass direction from a single image by bayesian inference," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 941–947 vol.2.

[7] A. Schwing, T. Hazan, M. Pollefeys, and R. Urtasun, "Efficient structured prediction for 3d indoor scene understanding," in *2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012*, ser. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2012, pp. 2815–2822.

[8] L. Del Pero, J. Bowdish, D. Fried, B. Kermgard, E. Hartley, and K. Barnard, "Bayesian geometric modeling of indoor scenes," in *2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012*, ser. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Oct. 2012, pp. 2719–2726, 2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2012 ; Conference date: 16-06-2012 Through 21-06-2012.

[9] W. Zhang, W. Zhang, and J. Gu, "Edge-semantic learning strategy for layout estimation in indoor environment," *CoRR*, vol. abs/1901.00621, 2019. [Online]. Available: http://arxiv.org/abs/1901.00621

[10] A. Mallya and S. Lazebnik, "Learning informative edge maps for indoor scene layout prediction," in *2015 International Conference on Computer Vision, ICCV 2015*, ser. Proceedings of the IEEE International Conference on Computer Vision. United States: Institute of Electrical and Electronics Engineers Inc., Feb. 2015, pp. 936–944,

15th IEEE International Conference on Computer Vision, ICCV 2015 ; Conference date: 11-12-2015 Through 18-12-2015.

[11] C.-Y. Lee, V. Badrinarayanan, T. Malisiewicz, and A. Rabinovich, "Roomnet: End-to-end room layout estimation," 2017.

[12] H. J. Lin and S.-H. Lai, "Deeproom: 3d room layout and pose estimation from a single image," in *Pattern Recognition*, S. Palaiahnakote, G. Sanniti di Baja, L. Wang, and W. Q. Yan, Eds. Cham: Springer International Publishing, 2020, pp. 719–733.

[13] Y. Ren, S. Li, C. Chen, and C.-C. J. Kuo, "A coarse-to-fine indoor layout estimation (cfile) method," in *Computer Vision – ACCV 2016*, S.-H. Lai, V. Lepetit, K. Nishino, and Y. Sato, Eds. Cham: Springer International Publishing, 2017, pp. 36–51.

[14] H. Zhao, M. Lu, A. Yao, Y. Guo, Y. Chen, and L. Zhang, "Physics inspired optimization on semantic transfer features: An alternative method for room layout estimation," 2017.

[15] C. Yan, B. Shao, H. Zhao, R. Ning, Y. Zhang, and F. Xu, "3d room layout estimation from a single rgb image," *IEEE Transactions on Multimedia*, vol. 22, no. 11, pp. 3014–3024, 2020.

[16] I. Kruzhilov, M. Romanov, D. Babichev, and A. Konushin, "Double refinement network for room layout estimation," in *Pattern Recognition*. Cham: Springer International Publishing, 2020, pp. 557–568.

[17] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," 2016.

[18] Y. Zhang, S. Song, P. Tan, and J. Xiao, "Panocontext: A whole-room 3d context model for panoramic scene understanding," in *Computer Vision – ECCV 2014*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds. Cham: Springer International Publishing, 2014, pp. 668–686.

[19] C. Zou, A. Colburn, Q. Shan, and D. Hoiem, "Layoutnet: Reconstructing the 3d room layout from a single rgb image," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 2051–2059.

[20] S.-T. Yang, F.-E. Wang, C.-H. Peng, P. Wonka, M. Sun, and H.-K. Chu, "Dula-net: A dual-projection network for estimating room layouts from a single rgb panorama," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 3363–3372.

[21] C. Sun, C.-W. Hsiao, M. Sun, and H.-T. Chen, "Horizonnet: Learning room layout with 1d representation and pano stretch data augmentation," in *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 1047–1056.

[22] C. Fernandez-Labrador, J. M. Facil, A. Perez-Yus, C. Demonceaux, J. Civera, and J. J. Guerrero, "Corners for layout: End-to-end layout recovery from 360 images," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1255–1262, 2020.

[23] J. Xiao, K. A. Ehinger, A. Oliva, and A. Torralba, "Recognizing scene viewpoint using panoramic place representation," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2012.

[24] I. Armeni, S. Sax, A. R. Zamir, and S. Savarese, "Joint 2d-3d-semantic data for indoor scene understanding," 2017.

[25] J. Zheng, J. Zhang, J. Li, R. Tang, S. Gao, and Z. Zhou, "Structured3d: A large photo-realistic dataset for structured 3d modeling," in *Proceedings of The European Conference on Computer Vision (ECCV)*, 2020.

[26] S. Stekovic, S. Hampali, M. Rad, S. D. Sarkar, F. Fraundorfer, and V. Lepetit, "General 3d room layout from a single view by render-and-compare," in *Computer Vision – ECCV 2020*. Cham: Springer International Publishing, 2020, pp. 187–203.

[27] H. Howard-Jenkins, S. Li, and V. A. Prisacariu, "Thinking outside the box: Generation of unconstrained 3d room layouts," in *ACCV*, 2018.

[28] C. Yang, J. Zheng, X. Dai, R. Tang, Y. Ma, and X. Yuan, "Learning to reconstruct 3d non-cuboid room layout from a single rgb image," in *WACV*, 2022.

[29] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "Scannet: Richly-annotated 3d reconstructions of indoor scenes," in *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

[30] A. Heyden, "Consistency and correction of line-drawings, obtained by projections of piecewise planar objects," in *European Conference on Computer Vision*. Springer, 1994, pp. 411–419.

[31] F. Kahl and A. Heyden, "Structure and motion from points, lines and conics with affine cameras," in *European Conference on Computer Vision*. Springer, 1998, pp. 327–341.

[32] K. Åström, A. Heyden, F. Kahl, and M. Oskarsson, "Structure and motion from lines under affine projections," in *Proceedings of the Seventh IEEE International Conference on Computer Vision*, vol. 1. IEEE, 1999, pp. 285–292.

[33] M. Oskarsson, A. Zisserman, and K. Åström, "Minimal projective reconstruction for combinations of points and lines in three views," *Image and Vision Computing*, vol. 22, no. 10, pp. 777–785, 2004.

[34] A. Bartoli and P. Sturm, "Structure-from-motion using lines: Representation, triangulation, and bundle adjustment," *Computer vision and image understanding*, vol. 100, no. 3, pp. 416–441, 2005.

[35] V. Larsson, K. Åström, and M. Oskarsson, "Efficient solvers for minimal problems by syzygy-based reduction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 820–829.

[36] B. Micusik and H. Wildenauer, "Structure from motion with line segments under relaxed endpoint constraints," *International Journal of Computer Vision*, vol. 124, no. 1, pp. 65–79, 2017.

[37] T. Lemaire and S. Lacroix, "Monocular-vision based slam using line segments," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, 2007, pp. 2791–2796.

[38] G.-S. Xia, J. Delon, and Y. Gousseau, "Accurate junction detection and characterization in natural images," *International journal of computer vision*, vol. 106, no. 1, pp. 31–56, 2014.

[39] N. Xue, S. Bai, F. Wang, G.-S. Xia, T. Wu, and L. Zhang, "Learning attraction field representation for robust line segment detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 1595–1603.

[40] N. Xue, S. Bai, F.-D. Wang, G.-S. Xia, T. Wu, L. Zhang, and P. H. Torr, "Learning regional attraction for line segment detection," *IEEE transactions on pattern analysis and machine intelligence*, 2019.

[41] Q. Meng, J. Zhang, Q. Hu, X. He, and J. Yu, *LGNN: A Context-Aware Line Segment Detector*. New York, NY, USA: Association for Computing Machinery, 2020, p. 4364–4372. [Online]. Available: https://doi.org/10.1145/3394171.3413784

[42] N.-G. Cho, A. Yuille, and S.-W. Lee, "A novel linelet-based representation for line segment detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1195–1208, 2018.

[43] K. Huang, Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma, "Learning to parse wireframes in images of man-made environments," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[44] Y. Sun, X. Han, K. Sun, B. Li, Y. Chen, and M. Li, "Sem-lsd: A learning-based semantic line segment detector," 2019.

[45] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision - 14th European Conference, ECCV 2016, Proceedings*, ser. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Germany: Springer Verlag, 2016, pp. 483–499.

[46] L. W. Beineke, "Characterizations of derived graphs," *Journal of Combinatorial Theory*, vol. 9, no. 2, pp. 129–135, 1970. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0021980070800199

Appendix

This is supplementary material for our paper ”Semantic Room Wireframe Detection from a Single View”. It further describes the data generation process and provides more images for qualitative evaluation.

Structured3D-SRW Data Generation

This section gives a brief explanation on how our dataset Structured3D-SRW, described in Section 4 of the paper, is generated. For further details we refer to the code.

First we define for a point $r = \begin{pmatrix} x_1 & x_2 & \dots & x_n \end{pmatrix} \in \mathbb{R}^n$ and its corresponding point $p = \begin{pmatrix} y_1 & y_2 & \dots & y_{n+1} \end{pmatrix} \in \mathbb{P}^n$ in homogeneous coordinates the mapping function

$$\mathbb{P}^n \to \mathbb{R}^n : \phi(p) = \begin{pmatrix} \frac{y_1}{y_{n+1}} & \frac{y_2}{y_{n+1}} & \dots & \frac{y_n}{y_{n+1}} \end{pmatrix}. \tag{6}$$

The task is to generate a set of junctions J_i connected by edges E_i for each image I_i. To simplify we will consider a single line segment $\hat{l} = \begin{pmatrix} \hat{p_1} & \hat{p_2} \end{pmatrix}$ with endpoints $\hat{p}_1, \hat{p}_2 \in \mathbb{P}^3$ s.t $\phi(\hat{p}_1), \phi(\hat{p}_2) \in \mathbb{R}^3$ are in the scene coordinate system. as well as the scene's defining set of planes, which have polygons $W = \{w_k, k = 1, ..., N\}$ and plane parameters

$$\hat{\Pi} = \left\{ \hat{\pi}_k = \begin{pmatrix} \hat{n}_k \\ \hat{d} \end{pmatrix} \middle| \, \|\hat{n}_k\| = 1, \phi(p) \in w_k \Rightarrow p^T \hat{\pi}_k = 0 \right\}, \tag{7}$$

where $\hat{n}_k \in \mathbb{R}^3$ is the normal vector.

For now we assume that the line segment is in front of the camera and inside the image. We start by transforming the planes and line segment to the camera centered coordinate system

$$\begin{aligned} &L = \begin{pmatrix} p_1 & p_2 \end{pmatrix} = T_i \hat{L}, \\ &\Pi = \left\{ \pi_k = T^{-1} \hat{\pi}_k \,\middle|\, \hat{\pi} \in \hat{\Pi} \right\}, \text{ where } T_i = \begin{pmatrix} R_i & t_i \\ 0 & 1 \end{pmatrix}. \end{aligned} \tag{8}$$

For each plane π_k we find where on the viewing ray for line segment endpoint p_j we have the plane. So if

$$a_j = -\frac{d_k}{\phi(p_j)^T n_k} \tag{9}$$

we know that $0 < a_j < 1$ if the point is occluded. Now we have four cases

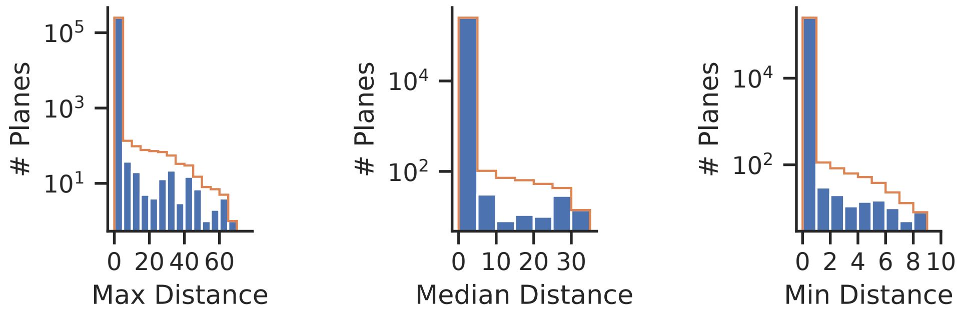


Figure 7: Histograms for the maximum, median and minimum distance from each plane to its junctions using the estimated plane equation.

- $a_1, a_2 \geq 1 \Rightarrow$ The entire line segment is in front of plane, we are done.
- $a_1, a_2 \leq 0 \Rightarrow$ The plane is behind the camera, we are done.
- $a_1, a_2 < 1 \Rightarrow$ The line segment is behind the plane, $L_b = L$.
- For all other cases a part of the line segment is in front of the plane. We form the line segment L_b which is behind the plane and L_f which is in front.

Now we can project L_b onto the plane π_k and take the 2D geometric difference to get a set of line segments

$$\hat{L}_w = \left(\begin{pmatrix} a_1 \\ a_2 \end{pmatrix} \odot L_b \right) - w_k, \tag{10}$$

where $\odot$ is elementwise multiplication. These are in the plane π_k so we so we find the corresponding line segments on L and denote them L_w. We then merge L_f with L_w and get the visible set of line segments which we transform back to the camera coordinate system and apply K_i to get the pixel coordinates. An overview of the algorithm is given in Algorithm 1. Please refer to the implementation code for details.

Plane estimation

In Section 4.1 we estimate new plane parameters that fit optimally with the plane junction. See Figure 7 for the histograms of distances from plane to junctions.

Algorithm 1: An overview of how the algorithm to annotate one scene works. For details and special cases please refer to the implementation code.

```
Data: Parameters of scene planes: $\hat{\mathbf{\Pi}}$, Semantic images: $\hat{\mathbf{I}}$, RGB images: $\mathbf{I}$, All
      plane junctions: $\mathbf{J}$, All scene lines: $\hat{\mathbf{L}}$,
Result: Set of visible line segments $\mathbf{L}_V^i$ for each image $i$
$W :=$ makePolygons($\hat{I}, \hat{\Pi}, J$);
for $I_i \in I$ do
    $\Pi := T_i^{-1}\hat{P}_i$;
    $\mathbf{L} := T_i\hat{\mathbf{L}}$;
    for $L \in L$ do
        $L :=$cutInFrontAndInImage($L$);
        if $L = \emptyset$ then
            continue;
        end
        $\mathbf{L}_V^i = \{L\}$ ;
        for $w_k \in W, \pi_k \in \Pi$ do
            $L_t := \emptyset$;
            for $L \in L_V^i$ do
                $L_f, L_b :=$ behindPlane($\pi_k, L$);
                $\mathbf{L}_p :=$ inPolygon($L_b$);
                $L_t := L_t \cup$ appendLine($L_p, L_f$);
            end
            $\mathbf{L}_V^i := L_t$;
        end
    end
end
```

Additional Images for Qualitative Comparison

Here is and extension to Section 7.3 of the paper, where we compared SRW-Net a Room Layout estimation algorithm [1]. For additional images from Structured3D-SRW, see Figure 8.

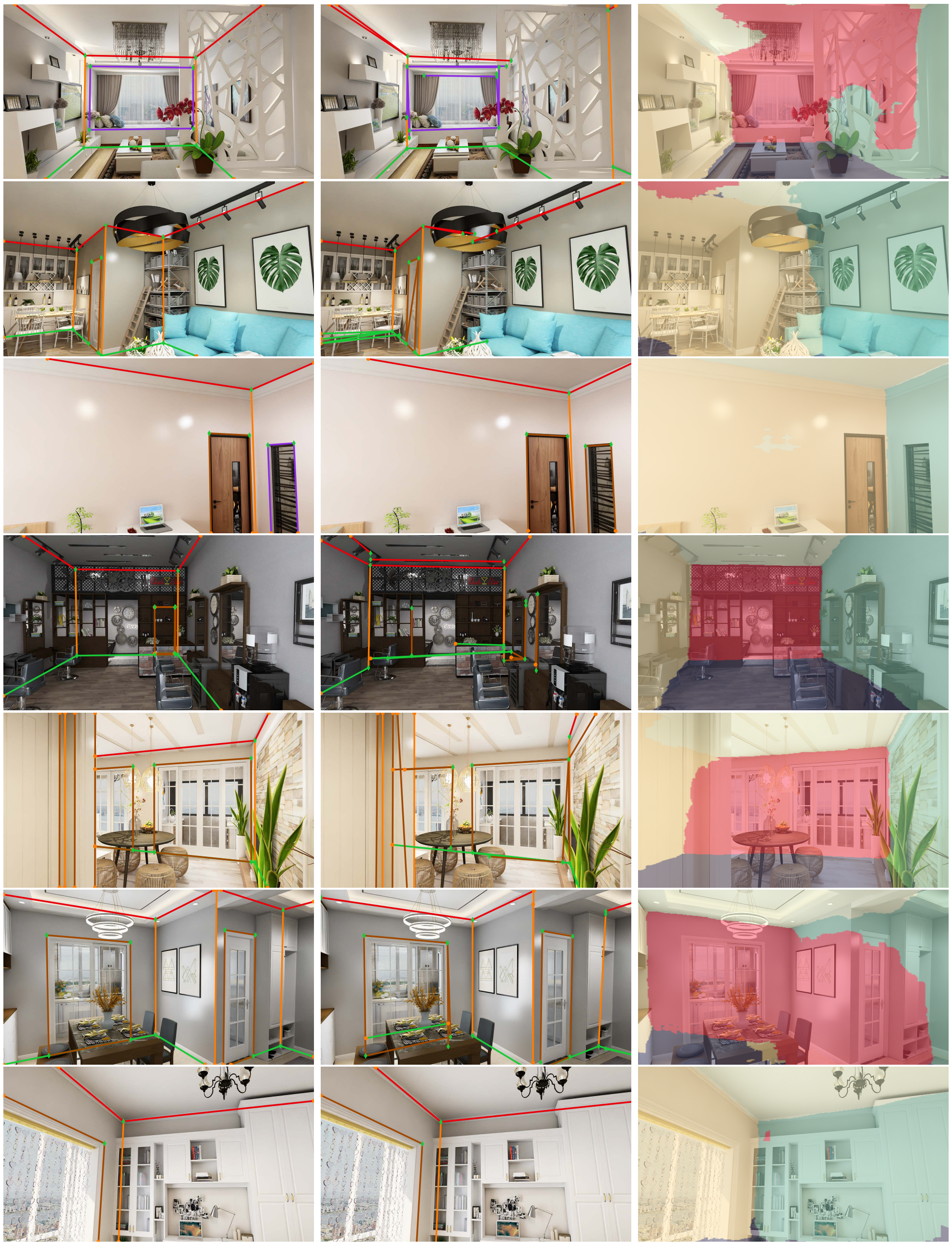


Figure 8: Results from the comparison with room layout estimation. On the left: images with ground truth semantic wireframe annotations. In the middle: the results from our method. To the right: room layout results from [1].

Paper IV

The Multi-View Geometry of Parallel Cylinders

Erik Tegler, Johanna Engman, David Gillsjö, Gabrielle Flood, Viktor Larsson, Magnus Oskarsson and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: In this paper we study structure from motion problems for parallel cylinders. Using sparse keypoint correspondences is an efficient (and standard) way to solve the structure from motion problem. However, point features are sometimes unavailable and they can be unstable over time and viewing conditions. Instead, we propose a framework based on silhouettes of quadric surfaces, with special emphasis on parallel cylinders. Such structures are quite common, e.g. trees, lampposts, pillars, and furniture legs. Traditionally, the projection of the center lines of such cylinders have been considered and used in computer vision. Here, we demonstrate that the apparent width of the cylinders also contains useful information for structure and motion estimation. We provide mathematical analysis of relative structure and relative motion tensors, which is used to develop a number of minimal solvers for simultaneously estimating camera pose and scene structure from silhouette lines of cylinders. These solvers can be used efficiently in robust estimation schemes, such as RANSAC. We use Sampson-approximation methods for efficient estimation using over-determined data and develop averaging techniques. We also perform synthetic accuracy and robustness tests and evaluate our methods on a number of real-world scenarios.

Keywords: Structure from motion, Parallel cylinders, Minimal Solvers, Sampson approximation

1 Introduction

Structure from motion (SfM) and simultaneous localization and mapping (SLAM) pipelines have a long history, and there exist several real-time systems for 3D reconstruction, e.g. [31, 43, 44]. These are often based on sparse point features extracted from images [27, 41]. Such systems are made possible by a long history of research results on the multi-view geometry of points, epipolar geometry, trifocal tensors, RANSAC, bundle adjustment, rotation averaging, etc. Sparse point features are efficient in terms of matching, tracking and optimization. However, there exists a number of limitations with such representations. Firstly, they carry little semantic information. Further processing of the scene, as extracting and recognizing objects, requires running additional algorithms. Secondly, sparse features are

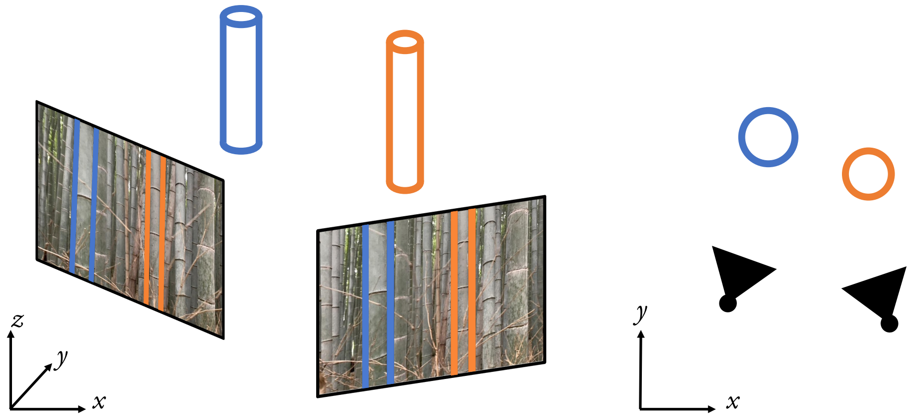


Figure 1: Multi-view geometry of parallel cylinders. *Left:* Each camera observes a set of parallel cylinders. Since the cylinders are parallel in 3D, the images can be rectified such that each projected cylinder is parallel in the image. *Right:* In the rectified coordinate system we cannot observe the translation of the camera along the cylinders (the z-axis). We can consider the remaining problem in 2D (the xy-plane).

not very stable over time, neither on a small, nor on a large time scale. If we want to return to a scene and use the previously created 3D reconstruction for localization or matching it can be difficult to match old features to new, due to changes in the scene environment such as lighting, season or weather condition [42]. Thirdly, sparse point features carry little geometric information. This motivates the use of additional features that inherently carry more semantic and geometric information about the scene. In many man-made environments, other geometric primitives can be used to describe the 3D geometry, e.g. planes, lines and conics [3, 4, 20, 37]. There has also been an increased interest in richer point features, for example, point plus scale or point plus directions [8, 23, 40].

In this paper we study structure from motion problems for parallel cylinders. Such cylinders appear in both natural (trees in a forest) or man-made (light-poles, fences, pillars) environments. Assuming that the cylinders are parallel makes it possible to reduce the problem from the case of 3D-to-2D-projections to the case of 2D-to-1D-projections. The parallel structures can be used to rectify the images, see Figure 1 (left). After rectification, the z-coordinates of both scene points and cameras are irrelevant and the problem reduces to a 2D-to-1D-projection problem, see Figure 1 (right). Traditionally, only the center lines of the parallel cylinders have been used for these kinds of problems. There is a relatively rich understanding of the geometry of the 2D-to-1D-projection case, for example [2, 33–35]. This paper can be viewed as an extension of such problems.

Related Work

Concerning geometric algorithms for conics, in [26] the authors introduce a method for estimating relative pose for objects that are rotationally symmetric. The methods are iterative and do not handle pose or SfM for multiple objects. Bundle adjustment for points has been extensively covered in many publications, see e.g. [46]. For lines and conics it is not as straightforward to define and use an appropriate and statistically sound error model. This problem has, however, been studied e.g. in [7, 18]. Pose of conics and quadrics has been studied for example in [10, 16, 28]. In [21, 38] it is shown that each projection of a conic or silhouette of a quadric provides two constraints for the two-view relative orientation problem. This is essentially the epipolar tangency constraint used in SfM for general silhouettes [5, 6]. For planar conics and circles there are several results on pose, homography estimation and relative pose, e.g. in [11, 12, 22, 29, 30]. Robust triangulation was investigated in [17]. The paper that is closest to our work is [32], in which the authors present a series of algorithms for pose, 3D reconstruction and structure from motion from cylinders. However, these algorithms do not work with minimal data and there are no robust estimation algorithms.

While line detections from cylinder outlines are assumed to be given in this paper, there are a number of relevant methods that may be used to automatically retrieve line segments from an image. Fast gradient based methods like LSD [14] and EDlines [1] are typically outperformed by neural network based methods. Many of these are trained on wireframe parsing datasets [9, 19, 50] consisting of connected line segments. Some methods like L-CNN [51], HAWP [49], SOLD2 [36] and SRW-Net [13] have multi-step pipelines which detects line segment endpoints and lines separately to later form a connected wireframe by matching. Others like AFM [48], M-LSD [15] and LETR [47] predict line segments directly.

Main Contribution

In this work we show how the structure from motion problem for multiple views of multiple parallel cylinders simplifies to the problem of 1D-views of circles in the plane. We completely characterize the two-view geometry, introduce matching tensors for both two-view relative motion and for two-circle relative structure and show that there is an interesting duality between pairs of cameras and pairs of circles. Furthermore, we derive fast minimal solvers and fast over-determined methods based on the Sampson approximation of the bundle adjustment errors and test our solvers in a structure from motion system and study how our methods work on both synthetic and real data.

2 Cylinder Geometry

We will now investigate which relative pose problems from images of conics that might be interesting and solvable. In order to do this we can study how many excess constraints

Table 1: The table shows different minimal relative pose problems. Many of these examples are both difficult to solve and not of so much practical use. In this paper we concentrate on the tractable minimal case in the last row.

Object type	Constraints	Object DoF	#Cameras	#Objects
Quadrics	9	5	2	5
Elliptic cylinders	4	7	2	5
Parallel Elliptic Cylinders	2	5	3	5
Parallel Circular Cylinders	2	3	2	2

$\mathcal{E}$ such problems have, i.e. the number of constraints minus the number of unknown variables. For a given problem we typically have

$$\mathcal{E} = \mathcal{A}mn - (\mathcal{B}m + \mathcal{C}n - \mathcal{D}), \tag{1}$$

where m denotes the number of images and n the number of objects and $\mathcal{A}, \mathcal{B}, \mathcal{C}$ and $\mathcal{D}$ denotes the degrees of freedom of the image object, camera, world object and coordinate system, respectively. In Table 1 the possible minimal cases are listed, i.e. where $\mathcal{E} = 0$. Many of these cases are both very difficult to solve and have limited practical use. Therefore we will in the remainder of this paper concentrate on the most interesting and tractable case, namely *parallel cylinders*, as illustrated in Figure 1.

We can without loss of generality assume that the cylinders are parallel to the world z-axis. The first step is to detect the silhouette lines of the cylinders, using e.g. [13, 36, 49, 51]. Assuming calibrated cameras it is relatively straightforward to robustly estimate the joint intersection point u_i of the lines in each image i. This point will be the projection of the point at infinity, at which the parallel lines in 3D meet. After applying an image homography R_i to each image, so that u_i maps to $R_i u_i = [0\ 1\ 0]$, all image lines will be parallel to the y-direction in the image plane, see Figure 1. This means that the problem can be formulated as a 2D problem of circles (in the $xy-$plane) projected into a reduced camera. In each reduced camera there are three degrees of freedom remaining, the camera center (x, y) in the $xy-$plane, and a single rotation (with angle θ) around the z-axis. Thus, for each image the camera can be parameterized using three parameters. Furthermore, in this case, the cylinders are circles in the xy-plane, and these can be parameterized using three parameters as well, the center point (u, v) and the radius r. For each cylinder, we observe its silhouette in the image. An illustration of this from above can be seen in Figure 2. In the xy-plane we use θ to denote the camera direction with respect to the coordinate system. The silhouette—seen from the camera at (x, y)—will now correspond to the angles to the left and right tangent point of the circle $c = (u, v, r)$, relative to the camera direction. This projection can be represented with two angles; first the angle α from the camera center to the center of the circle relative to the camera orientation, and second the angular half-width β of the image of the circle. To project a circle $c = (u, v, r)$ into a camera $p = (x, y, \theta)$ we

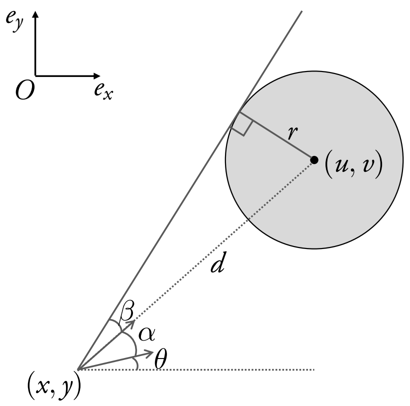


Figure 2: The figure shows the image (angles α, β) of a circle (u, v, r) as seen from a camera (x, y, θ). The tangents to the circle from the point (x, y) form angles $(\alpha + \beta), (\alpha - \beta)$ to the viewing angle θ. The depth d, the radius r and angle β are related through $d \sin(\beta) = r$.

compute

$$\alpha(p, c) = \text{atan2}(u - x, v - y) - \theta, \tag{2}$$

$$\beta(p, c) = \text{asin}\left(\frac{r}{d}\right), \tag{3}$$

where $d = \sqrt{(u-x)^2 + (v-y)^2}$ is the distance between the circle center and the camera center. Note that $\alpha(p, c)$ does not depend on the radius r, and similarly $\beta(p, c)$ does not depend on the camera orientation θ.

It is possible to find the circle structure and the motion of the cameras using only one type of measurements. If α_{ij} is used the problem is essentially an ordinary structure from motion problem of 2D points projected to a 1D retina which has been studied in several papers, for example [3, 39]. The minimal problems are three views of five points—which has two solutions—and four views of four points—which also has two solutions [2]. The problem of retaining the structure and motion using only the measurements β_{ij} is less well studied. Given measurements of β it is possible to calculate the distances d_{ij} between all circle centers and all camera centers. These expressions also involve the unknown radii. If the radii were all known this would be the node calibration problem for time-of-arrival measurements, which has been studied extensively, e.g. in [24]. The minimal problems are three views of three points, which has eight solutions, [45]. However since the radii are also unknown, it is possible to use more measurements to first calculate the radii and then solve the time-of-arrival problem.

In this paper we study different aspects of the structure from motion problem when we have both types of measurements mentioned above. In other words, we study the problem of determining both structure $c_1, \ldots, c_n$ and motion $p_1, \ldots, p_m$ from the image measurements $(\bar{\alpha}_{ij}, \bar{\beta}_{ij}), i = 1, \ldots, m, j = 1, \ldots, n$. Often some image measurements are missing, so we introduce the index set $\mathcal{I}$ of (i, j) if cylinder j is visible in image i. In the end we seek the solution to an optimization problem (bundle adjustment)

$$\min_{p,c} \sum_{(i,j)\in\mathcal{I}} L\left(\frac{\alpha(p_i, c_j) - \bar{\alpha}_{ij}}{\sigma_{\alpha_{ij}}}\right) + \sum_{(i,j)\in\mathcal{I}} L\left(\frac{\beta(p_i, c_j) - \bar{\beta}_{ij}}{\sigma_{\beta_{ij}}}\right), \tag{4}$$

where σ denotes estimated standard deviations of measurements. Here we use the squared loss $L(z) = z^2$, but it is possible to use L^1 loss, truncated L^2 or L^1 or any other robust loss.

3 The Multi-View Geometry of Parallel Cylinders

One approach to solve the structure from motion problem is to formulate the problem as a system of polynomial equations. We can then use an existing automatic solver generator [25] to generate a solver for these equations. Our original system of equations (2) and (3) can be reformulated using the addition rule of tangents, and the change of variables

$$s_i = \tan(\theta_i), \tag{5}$$

to get

$$\begin{cases} (u_j - x_i)(\tan\alpha_{ij} + s_i) - (v_j - y_i)(1 - \tan\alpha_{ij} s_i) = 0, \\ \sin^2(\beta_{ij})((u_j - x_i)^2 + (v_j - y_i)^2) - r_j^2 = 0, \end{cases} \tag{6}$$

with variables $\{x_1, x_2, y_1, y_2, s_1, s_2, u_1, u_2, v_1, v_2, r_1, r_2\}$. This is a system of eight polynomial equations in twelve unknowns, of which four can be fixed by a suitable choice of coordinate system. An efficient solver can be produced using the system [25], that gives at most eight (possibly complex) solutions. However, as will be explained in the next sections, interesting alternative solvers can be constructed using a relative motion and/or relative structure tensor, also these will avoid some symmetries in the solution structure.

3.1 The Relative Motion Tensor $\mathcal{T}$

The fundamental matrix encodes the epipolar geometry of two 3D-to-2D perspective cameras. The matrix provides constraints on which measurements are consistent with the particular camera configuration. In this section we derive the analogous matching tensor for the particular 2D-to-1D setting that we consider. This new tensor encodes the relative position of two cameras and yields a constraint on each cylinder correspondence, similar to the epipolar constraint.

The relative motion of two cameras can be characterized by two angles, θ_1 of the first camera and θ_2 of the second camera. This relative motion can be understood as normalizing the

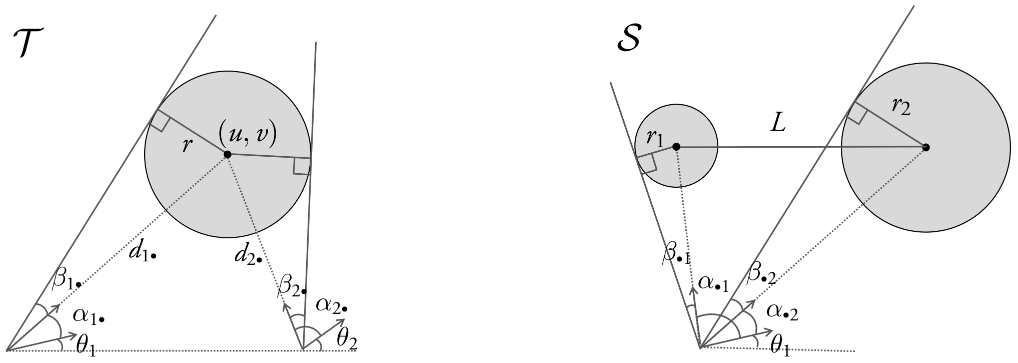


Figure 3: *Left:* The relative motion tensor $\mathcal{T}$ describing the constraint from a circle-correspondence in two-views. *Right:* The relative structure tensor $\mathcal{S}$ describing the constraint between two circles observed in one view.

coordinate system by placing the first camera in $(0, 0)$ and the second in $(1, 0)$. Note that we can w.l.o.g. assume the baseline between the cameras to be of unit length since the scale is not observable.

One circle c seen in these two images gives four measurements: $(\alpha_{1\bullet}, \beta_{1\bullet})$ from the first camera and $(\alpha_{2\bullet}, \beta_{2\bullet})$ from the second camera, where the subscript dot index indicates that the angle is w.r.t. some circle.

Theorem 8.1. Two views $(\alpha_{1\bullet}, \beta_{1\bullet}, \alpha_{2\bullet}, \beta_{2\bullet})$ of one circle is only consistent with a relative motion (θ_1, θ_2) if

$$\begin{bmatrix} \cos(\alpha_{1\bullet})\sin(\beta_{2\bullet}) \\ \sin(\alpha_{1\bullet})\sin(\beta_{2\bullet}) \\ -\cos(\alpha_{2\bullet})\sin(\beta_{1\bullet}) \\ -\sin(\alpha_{2\bullet})\sin(\beta_{1\bullet}) \end{bmatrix}^T \begin{bmatrix} \sin(\theta_1) \\ \cos(\theta_1) \\ \sin(\theta_2) \\ \cos(\theta_2) \end{bmatrix} = 0. \tag{7}$$

Proof. Consider two cameras with relative motion (θ_1, θ_2) and measurements $(\alpha_{1\bullet}, \beta_{1\bullet}, \alpha_{2\bullet}, \beta_{2\bullet})$ as illustrated in Figure 3 (left). If the measurements are consistent with a circle with parameters (u, v, r), then consider the triangle of the two camera centers and the circle center. Using the law of sines we have that

$$\frac{\sin(\theta_1 + \alpha_{1\bullet})}{d_{2\bullet}} = \frac{\sin(\theta_2 + \alpha_{2\bullet})}{d_{1\bullet}}. \tag{8}$$

From (3) we get $d_{1\bullet} = \frac{r}{\sin\beta_{1\bullet}}$ and $d_{2\bullet} = \frac{r}{\sin\beta_{2\bullet}}$. Inserting this into (8) gives

$$\frac{\sin(\theta_1 + \alpha_{1\bullet})\sin(\beta_{2\bullet})}{r} = \frac{\sin(\theta_2 + \alpha_{2\bullet})\sin(\beta_{1\bullet})}{r}, \tag{9}$$

or $\sin(\theta_1 + \alpha_{1\bullet})\sin(\beta_{2\bullet}) - \sin(\theta_2 + \alpha_{2\bullet})\sin(\beta_{1\bullet}) = 0$, which can be rewritten as

$$\begin{gathered} \sin(\theta_1)\cos(\alpha_{1\bullet})\sin(\beta_{2\bullet}) + \cos(\theta_1)\sin(\alpha_{1\bullet})\sin(\beta_{2\bullet}) - \\ \sin(\theta_2)\cos(\alpha_{2\bullet})\sin(\beta_{1\bullet}) - \cos(\theta_2)\sin(\alpha_{2\bullet})\sin(\beta_{1\bullet}) = 0, \end{gathered} \tag{10}$$

which is the same as (7), hence, this concludes the proof. □ □

We will call

$$\mathcal{T}(\theta_1, \theta_2) = \begin{bmatrix} \sin(\theta_1) \\ \cos(\theta_1) \\ \sin(\theta_2) \\ \cos(\theta_2) \end{bmatrix} \in \mathbb{P}^3 \tag{11}$$

the **relative motion tensor**. It is a projective entity that has one non-linear homogeneous constraint

$$\mathcal{T}_1^2 + \mathcal{T}_2^2 = \mathcal{T}_3^2 + \mathcal{T}_4^2, \tag{12}$$

where subscript index denotes the element number. This can be rewritten as $\mathcal{T}^T D \mathcal{T} = 0$ for a suitable matrix D. When extracting θ_1 and θ_2 from the vector it might be natural to choose the scale of the vector so that $|\mathcal{T}| = 2$, since then $\mathcal{T}_1^2 + \mathcal{T}_2^2 = 1$ and $\mathcal{T}_3^2 + \mathcal{T}_4^2 = 1$.

Minimal Solver Using Relative Motion Tensor

Theorem 8.1 can be used to estimate the relative motion of two cameras using two views of two circles by first forming the matrix

$$M = \begin{bmatrix} \cos(\alpha_{11})\sin(\beta_{21}) & \sin(\alpha_{11})\sin(\beta_{21}) & -\cos(\alpha_{21})\sin(\beta_{11}) & -\sin(\alpha_{21})\sin(\beta_{11}) \\ \cos(\alpha_{12})\sin(\beta_{22}) & \sin(\alpha_{12})\sin(\beta_{22}) & -\cos(\alpha_{22})\sin(\beta_{12}) & -\sin(\alpha_{22})\sin(\beta_{12}) \end{bmatrix}. \tag{13}$$

We know that the relative motion tensor should be orthogonal to the rows in M, i.e. that $M\mathcal{T} = 0$. Using singular value decomposition of M we can parameterize all vectors orthogonal to the rows of M as

$$\mathcal{T}(z) = B_1 z_1 + B_2 z_2 = Bz, \tag{14}$$

where B is the basis of the null space. Inserting this into the quadratic constraint $\mathcal{T}^T D \mathcal{T}$ gives a homogeneous second degree equation

$$\mathcal{T}^T D \mathcal{T} = (Bz)^T D B z = z^T B^T D B z = z^T A z = 0, \tag{15}$$

in z. Equation (15) is a quadratic function in one projective variable, $z \in \mathbb{P}^1$.

For each of the two solutions of z, we can backtrack, i.e. first calculate the corresponding relative motion vector, from that obtain the camera angles

$$\theta_1 = \operatorname{atan2}(\mathcal{T}_1, \mathcal{T}_2), \quad \theta_2 = \operatorname{atan2}(\mathcal{T}_3, \mathcal{T}_4), \tag{16}$$

and finally, compute the circles.

3.2 The Relative Structure Tensor $\mathcal{S}$

In the previous section we derived a constraint on a single cylinder observed in two cameras. Interestingly, there is a symmetric variant of this problem, where we instead consider one camera observing two cylinders. In this case it is possible to derive an analogous constraint, however this time on the radii and distance between the two cylinders. Again this is formulated in terms of a matching tensor, which similarly to the relative motion tensor $\mathcal{T}$, can be used to formulate a fast minimal solver.

The relative structure of two circles can be characterized by the radius r_1 of the first circle, the radius r_2 of the second circle, and the distance L between the two circles. Since the scale is a gauge freedom, the vector (r_1, r_2, L) can be scaled arbitrarily. One view, p, of this circle-pair gives four measurements: $(\alpha_{\bullet 1}, \beta_{\bullet 1})$ to the first circle and $(\alpha_{\bullet 2}, \beta_{\bullet 2})$ to the second circle, subscript dot is now denoting an arbitrary camera.

Theorem 8.2. One view $(\alpha_{\bullet 1}, \beta_{\bullet 1}, \alpha_{\bullet 2}, \beta_{\bullet 2})$ is only consistent with a two-circle-configuration with relative structure (r_1, r_2, L) if

$$\begin{bmatrix} 1 & \frac{-1}{\sin^2(\beta_{\bullet 1})} & \frac{-1}{\sin^2(\beta_{\bullet 2})} & \frac{2\cos(\alpha_{\bullet 2}-\alpha_{\bullet 1})}{\sin(\beta_{\bullet 1})\sin(\beta_{\bullet 2})} \end{bmatrix} \begin{bmatrix} L^2 \\ r_1^2 \\ r_2^2 \\ r_1 r_2 \end{bmatrix} = 0. \tag{17}$$

Proof. The geometric situation is illustrated in Figure 3 (right). We will use the law of cosines on the triangle formed by the camera center and the centers of the two circles. This gives

$$L^2 = d_{\bullet 1}^2 + d_{\bullet 2}^2 - 2 d_{\bullet 1} d_{\bullet 2} \cos(\alpha_{\bullet 2} - \alpha_{\bullet 1}). \tag{18}$$

From (3) we get $d_{\bullet 1} = \frac{r_1}{\sin \beta_{\bullet 1}}$ and $d_{\bullet 2} = \frac{r_2}{\sin \beta_{\bullet 2}}$. Inserting this in (18) gives

$$L^2 - r_1^2 \frac{1}{\sin^2 \beta_{\bullet 1}} - r_2^2 \frac{1}{\sin^2 \beta_{\bullet 2}} + r_1 r_2 \frac{2\cos(\alpha_{\bullet 2} - \alpha_{\bullet 1})}{\sin \beta_{\bullet 1} \sin \beta_{\bullet 2}} = 0, \tag{19}$$

which is what we get from the matrix multiplication in Equation (8.2). This concludes the proof. □ □

We will call

$$\mathcal{S}(r_1, r_2, L) = \begin{bmatrix} L^2 \\ r_1^2 \\ r_2^2 \\ r_1 r_2 \end{bmatrix} \in \mathbb{P}^3 \tag{20}$$

the **relative structure tensor**. It is a projective entity that has one quadratic homogeneous constraint

$$\mathcal{S}_2\mathcal{S}_3 - \mathcal{S}_4^2 = 0. \tag{21}$$

where again subscript index denotes the element number. This can be rewritten as $\mathcal{S}^T C \mathcal{S} = 0$ for a suitable matrix C. Notice that for real two-circle configurations, we would normally assume that $r_1 > 0, r_2 > 0$, and $L > 0$, which means that we would have all elements of $\mathcal{S}$ positive.

Minimal Solver Using Relative Structure Tensor

Theorem 8.2 can be used to estimate the relative structure of two circles using two views of two circles by simply forming the matrix

$$M = \begin{bmatrix} 1 & \frac{-1}{\sin^2(\beta_{11})} & \frac{-1}{\sin^2(\beta_{12})} & \frac{2\cos(\alpha_{12}-\alpha_{11})}{\sin(\beta_{11})\sin(\beta_{12})} \\ 1 & \frac{-1}{\sin^2(\beta_{21})} & \frac{-1}{\sin^2(\beta_{22})} & \frac{2\cos(\alpha_{22}-\alpha_{21})}{\sin(\beta_{21})\sin(\beta_{22})} \end{bmatrix}. \tag{22}$$

Similar to before, the relative structure tensor should be orthogonal to M, giving the equation $M\mathcal{S} = 0$. This is solved analogously to the relative motion tensor problem presented in Equations (13)-(16).

4 Circle-Based Structure from Motion in 2D

In this section we present a framework for reconstructing both cameras and parallel cylinders. One of the difficult parts of a structure from motion estimation problem is that of identifying inliers in the feature detections, finding correspondences, and obtaining an initial estimate of the parameters. We use our proposed solvers in a RANSAC bootstrapping framework to find initial estimates of the structure and motions, and at the same time we estimate the inlier set.

Non-Minimal Estimators

The relative motion and relative structure tensors can be used to estimate structure and motion for the minimal cases as shown in Sections 3.1 and 3.2. They can also be used for fast estimation of structure from motion in the over-determined case. This works similar for both types of tensors, but we exemplify this with the relative motion tensor.

The relative motion tensor constraint can be used to linearly estimate $\mathcal{T}$ using $n \geq 3$ cylinders seen in two views by adding rows to the matrix M in (13) and finding the vector

$\mathcal{T}$ that minimize $|M\mathcal{T}|$ under the constraint $|\mathcal{T}| = 1$. This is done using singular value decomposition. From this it is again possible to calculate (θ_1, θ_2) and then finally calculate the cylinders. One advantage of estimating relative motion by minimizing the sum of the square of the residuals $M\mathcal{T}$ is that M only depends on image data and can be computed once and reused during the optimization. Another advantage is that $\mathcal{T}$ encodes relative motion, but the scene structure of the circles is not needed.

Sampson Approximation

The estimate from the previous section, is based on minimizing an algebraic error $|M\mathcal{T}|$, which is different from the bundle adjustment error (4) and thus, although fast, the result is sub-optimal. A better approximation to the bundle adjustment estimate can be obtained by considering the problem of adjusting the image data as little as possible so that the modified data matrix $\tilde{M}$ solves $\tilde{M}\mathcal{T} = 0$. This would solve the bundle adjustment problem, i.e. minimizing (4). If the residuals $M\mathcal{T}$ are small, we can approximate the closest distance to the manifold $\tilde{M}\mathcal{T} = 0$. For each residual this distance is approximated by $\frac{|M_i\mathcal{T}|}{|\nabla M_i\mathcal{T}|}$. Thus by weighting row M_i with the weight $w_i = \frac{1}{|\nabla M_i\mathcal{T}|}$ a better approximation to the bundle adjustment estimate can be obtained. The smaller the residuals are, the better the approximation is. The approximation is, however, only valid for relative motion tensors on the manifold $\mathcal{T}^T D\mathcal{T} = 0$.

Extending to a Complete Structure from Motion Pipeline

Using the tools from this paper it is possible to design structure from motion estimation systems for the general case of m views of n cylinders in the presence of noise, outliers and missing data. We have experimented with several variants, e.g. (i) **primal method** - pairwise estimation of relative motion followed by rotation and then translation averaging; (ii) **dual method** - pairwise estimation of relative structure, followed by radius averaging and multi-dimensional scaling; and (iii) **incremental method** - incremental robust structure from motion, starting with two-view RANSAC, followed by extensions to more views and cylinders through robust pose and robust triangulation. All these approaches allow for final adjustment of the parameters through non-linear optimization, see (4), often referred to as bundle adjustment.

5 Experimental Validation

In the following sections we experimentally evaluate the proposed solvers. In Section 5.1 we study both the numerical stability of the solvers and also the noise sensitivity of the problem.

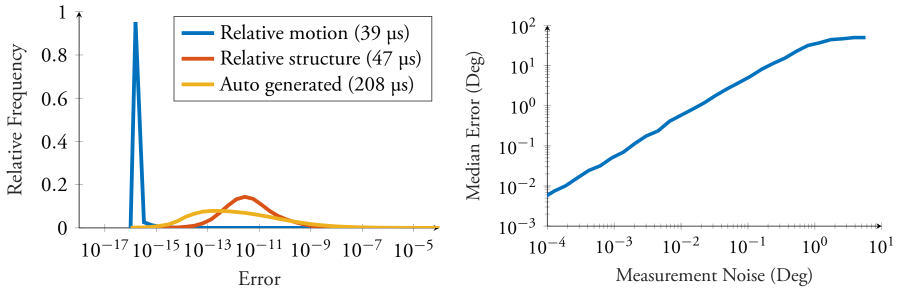


Figure 4: *Left:* Numerical stability of solvers. Runtimes of Matlab solvers given in legend. *Right:* Stability of the problem with respect to noise.

Finally, in Section 5.2, we show a series of experiments where we apply the method to real data.

5.1 Simulated Data

We evaluate the numerical stability of the solvers by studying how large reprojection errors we get when the solver is applied to a problem without any noise. In Figure 4 (left) a histogram of these reprojection errors is shown, for each of our three proposed solvers from Section 3. The first one is using the relative motion tensor and the second one is using the relative structure tensor. The third one is auto generated using [25] with eight equations (6) and eight unknowns. All three solvers are fast and provide solutions with small errors. Of the three the one based on relative motion is best in terms of speed and accuracy. In the right part of Figure 4 we show how sensitive the problem is to measurement noise. This was done by simulating a large number of problems consisting of two cameras observing two circles. We then solved for the camera viewing directions. The median error in this viewing direction was then plotted against the standard deviation of the normal distributed noise added to the α and β measurements. The graph shows that the estimated viewing directions have roughly 50 times larger median errors compared to the noise of the input data. The randomly generated problems were created by setting the two camera centers at $(-5, 0)$ and $(5, 0)$ and the cameras viewing direction were sampled uniformly from the interval $(\pi/4, 3\pi/4)$. Then the two circles' x-coordinates, y-coordinates and radii were sampled uniformly from $(-20, 20)$, $(5, 15)$ and $(0.2, 0.7)$ respectively.

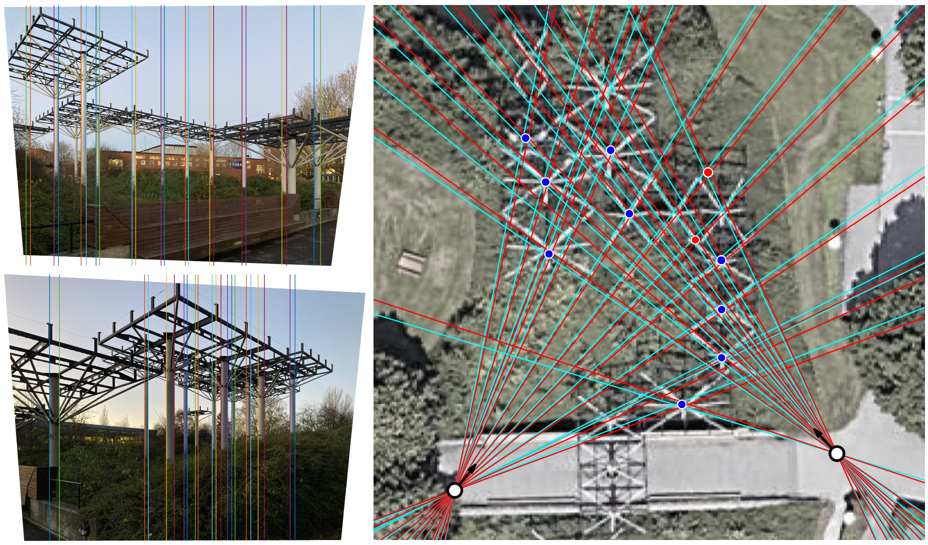


Figure 5: *Left:* two images with reprojected cylinder silhouettes, using the estimated solution. *Right:* the reconstruction registered to an top view. Also shown are the back-projected left and right silhouette lines in cyan and red. Note that there are no apparent image point correspondences in the two input images.

5.2 Real Data

Fountain Experiments

In Figure 5 (left) two images of a fountain scene are shown. We extracted silhouette lines manually (13 and 11 respectively). The images were then rectified, using the focal length from the EXIF-data of the images. We do not assume any known correspondences in this experiment. Note also that due to the extreme change in viewpoint, there are actually no point correspondences available, so a standard SfM pipeline such as e.g. COLMAP[43] will fail. We ran our relative motion solver, sampling exhaustively all combinations of two-cylinder correspondences. For each solution we checked how many inlier measurements we got in total, with the constraint that each cylinder can match at most one cylinder in the other image. This constraint was enforced by solving an assignment problem optimally for each solution. The resulting reconstruction is shown in Figure 5 (right), overlaid on a top-view aerial image of the scene. The lines to the left in the input images are the reprojected lines from the estimated solution. Since the cylinder widths are quite small with respect to the depths, and since we do not have any tentative correspondences, there are a number of possible (false) correspondences that give small reprojection errors. In the final reconstruction two cylinders (marked in red) have wrong correspondences, however, the camera estimates are reasonable.

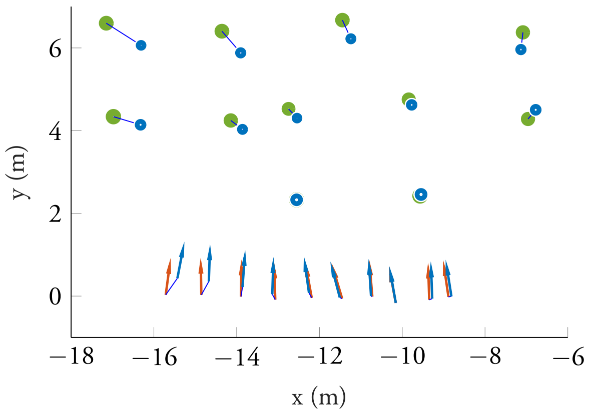


Figure 6: The forest reconstruction from a top view. The green circles are the ground truth trees position, the red arrows are the ground truth camera position and the blue circles and arrows are the solutions given by our system.

Forest Experiments

In this experiment we show how the solvers handle real data of approximately parallel cylinders, given from trees in a forest. The ground truth cameras and cylinder models were provided by a dedicated SLAM system. This system only detects a fraction of the trees so in addition to the trees not being perfect cylinders the lack of tracked cylinders constitutes an additional difficulty for our solvers. The cylinder lines were computed from manually labeled trunk silhouettes in ten images. Using our incremental SfM system, explained in Section 4, we estimate the trees and the cameras. The results are shown in Figure 6, where the trees are shown as circles and the cameras as arrows. The estimated cameras and trees are displayed in blue, while ground truth is plotted in red and green, respectively. From experience we know that the ground truth cameras in this dataset are much more reliable than the ground truth tree positions and therefore we evaluate this experiment only on the cameras. In this case the mean error for the translation of the cameras is 0.1228 m and the mean error for the rotation of the cameras is 1.3452 degrees. In Figure 7 (top) the reconstruction of a pair of cameras is shown together with one frame with the detected cylinders marked. Overall, considering that the trees are only approximately parallel cylinders, the results look very good.

Column Experiment

In this experiment, we will test the **dual** approach of first estimating a set of cylinders. In this often occurring use-case, we have a fixed structure of a set of cylinders—the columns of a building—as depicted in the bottom of Figure 7. We (manually) extract the silhouette lines in a number of images. Since the structure is fixed, and part of the building, the order

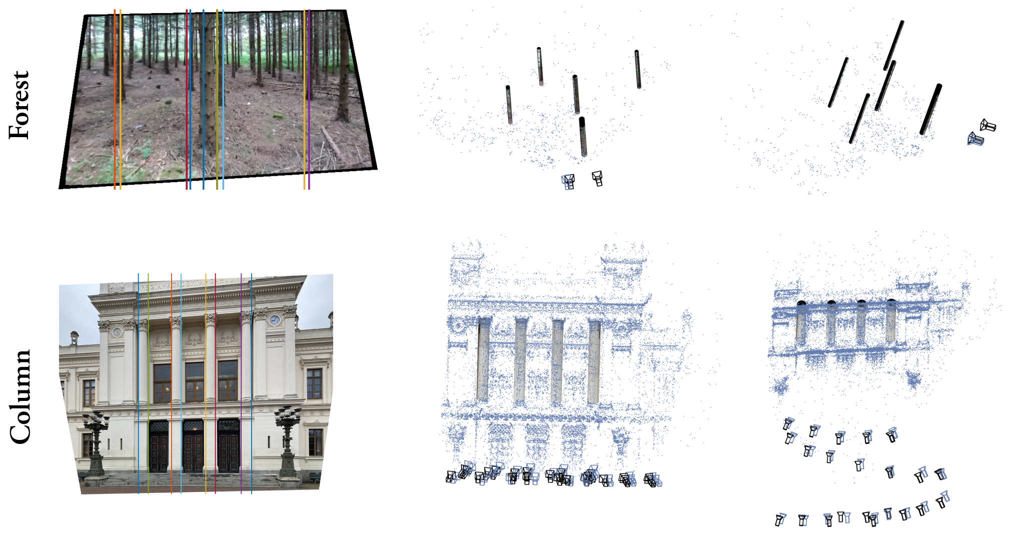


Figure 7: Qualitative reconstructions for the Forest (top) and Column (bottom) datasets.

will be the same in all images. This means that the correspondence problem of the cylinders across images is solved. We sample all six possible pairs of cylinders, and use the relative structure solver to estimate the cylinder geometry, using RANSAC over the 23 images for each pair. From the pairwise estimated cylinders we solve for a global estimate in a least squares sense. The camera poses are then linearly estimated followed by bundle adjustment over all cylinders and camera poses. We have also compared with a reconstruction using COLMAP. At the bottom of Figure 7 the reconstruction is shown, for two different viewpoints, with the estimated poses from COLMAP in blue. We have registered our reconstruction (setting the height to zero in all camera poses), and the results are also shown in the figure (with the estimated poses in black).

6 Conclusion

We have provided theory for the minimal structure from motion problem of two parallel cylinders in two views. We studied this both as a two-view relative motion and as a two-circle relative structure problem, using relative motion and structure tensors which, respectively, describe constraints for these 2D-to-1D mappings in a similar way as the fundamental matrix does for two 3D-to-2D cameras. We developed several minimal solvers and suggest how these can be used in a complete structure from motion pipeline and how to optimize the found solutions in real settings with noise and outlier measurements. These methods were evaluated, both on synthetic and real data. The variety of experiments conducted show both robustness of our solvers but also their potential in a system setting.

Previous solvers and algorithms that solve similar problems demand more data points in each observation, and—to the best of our knowledge—there are no similar solvers to compare with ours.

References

[1] Akinlar, C., Topal, C.: Edlines: Real-time line segment detection by edge drawing (ed). In: Proc. International Conference on Image Processing (ICIP). pp. 2837–2840 (2011). https://doi.org/10.1109/ICIP.2011.6116138

[2] Åström, K., Oskarsson, M.: Solutions and ambiguities of the structure and motion problem for 1d retinal vision. Journal of Mathematical Imaging and Vision **12**(2), 121–135 (2000)

[3] Åström, K.: Invariancy Methods for Points, Curves and Surfaces in Computational Vision. Ph.D. thesis, Lund Unicersiry (1996)

[4] Åström, K.: Using combinations of points, lines and conics to estimate structure and motion. In: Proc. Symposium on Image Analysis. pp. 61–64. SSBA (1998)

[5] Åström, K., Cipolla, R., Giblin, P.J.: Generalised epipolar constraints. International Journal of Computer Vision **33**, 51–72 (1999)

[6] Åström, K., Kahl, F.: Motion estimation in image sequences using the deformation of apparent contours. IEEE Transactions on Pattern Analysis and Machine Intelligence **21**(2), 114–127 (1999)

[7] Åström, K., Kahl, F., Heyden, A., Berthilsson, R.: A statistical approach to structure and motion from image features. In: Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR). pp. 929–936. Springer (1998)

[8] Barath, D., Polic, M., Förstner, W., Sattler, T., Pajdla, T., Kukelova, Z.: Making affine correspondences work in camera geometry computation. In: Proc. European Conference on Computer Vision (ECCV). pp. 723–740. Springer (2020)

[9] Cho, N.G., Yuille, A., Lee, S.W.: A novel linelet-based representation for line segment detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **40**(5), 1195–1208 (2018). https://doi.org/10.1109/TPAMI.2017.2703841

[10] De Ma, S.: Conics-based stereo, motion estimation, and pose determination. International journal of computer vision **10**(1), 7–25 (1993)

[11] Frosio, I., Alzati, A., Bertolini, M., Turrini, C., Borghese, N.A.: Linear pose estimate from corresponding conics. Pattern recognition **45**(12), 4169–4181 (2012)

[12] Frosio, I., Turrini, C., Alzati, A.: Camera re-calibration after zooming based on sets of conics. The Visual Computer **32**(5), 663–674 (2016)

[13] Gillsjö, D., Flood, G., Åström, K.: Semantic room wireframe detection from a single view. In: Proc. International Conference on Pattern Recognition (ICPR). pp. 1886–1893. IEEE (2022). https://doi.org/10.1109/ICPR56361.2022.9956252

[14] Grompone von Gioi, R., Jakubowicz, J., Morel, J.M., Randall, G.: Lsd: A fast line segment detector with a false detection control. IEEE Transactions on Pattern Analysis and Machine Intelligence **32**(4), 722–732 (2010). https://doi.org/10.1109/TPAMI.2008.300

[15] Gu, G., Ko, B., Go, S., Lee, S.H., Lee, J., Shin, M.: Towards light-weight and real-time line segment detection. In: Proc. Conference on Artificial Intelligence (AAAI). pp. 726–734 (2022)

[16] Gummeson, A., Engman, J., Åström, K., Oskarsson, M.: Fast and efficient minimal solvers for quadric based camera pose estimation. In: Proc. International Conference on Pattern Recognition (ICPR). pp. 3973–3979. IEEE (2022)

[17] Gummeson, A., Oskarsson, M.: Robust and accurate cylinder triangulation. In: Proc. Scandinavian Conference on Image Analysis (SCIA). Springer (2023)

[18] Hanek, R., Navab, N., Appel, M.: Yet another method for pose estimation: A probabilistic approach using points, lines, and cylinders. In: Proc. Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). vol. 2, pp. 544–550. IEEE (1999)

[19] Huang, K., Wang, Y., Zhou, Z., Ding, T., Gao, S., Ma, Y.: Learning to parse wireframes in images of man-made environments. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 626–635. IEEE (2018)

[20] Kahl, F., Heyden, A.: Structure and motion from points, lines and conics with affine cameras. In: Proc. European Conference on Computer Vision (ECCV). pp. 327–341. Springer (1998)

[21] Kahl, F., Heyden, A.: Using conic correspondences in two images to estimate the epipolar geometry. In: Proc. International Conference on Computer Vision (ICCV). pp. 761–766. IEEE (1998)

[22] Kaminski, J., Shashua, A.: Multiple view geometry of algebraic curves. International journal of computer vision (2003)

[23] Kuang, Y., Åström, K.: Pose estimation with unknown focal length using points, directions and lines. In: Proc. International Conference on Computer Vision (ICCV). pp. 529–536. IEEE (2013)

[24] Kuang, Y., Burgess, S., Torstensson, A., Åström, K.: A complete characterization and solution to the microphone position self-calibration problem. In: Proc. of International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3875–3879. IEEE (2013)

[25] Larsson, V., Åstrom, K., Oskarsson, M.: Efficient solvers for minimal problems by syzygy-based reduction. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 820–829. IEEE (2017)

[26] Liu, C., Hu, W.: Real-time geometric fitting and pose estimation for surface of revolution. Pattern Recognition **85**, 90–108 (2019)

[27] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International journal of computer vision **60**, 91–110 (2004)

[28] Ma, S., Li, L.: Ellipsoid reconstruction from three perspective views. In: Proc. International Conference on Pattern Recognition (ICPR). vol. 1, pp. 344–348. IEEE (1996)

[29] Mei, J., Zhang, D., Ding, Y.: Monocular vision for pose estimation in space based on cone projection. Optical Engineering **56**(10), 103108 (2017)

[30] Mudigonda, P.K., Jawahar, C., Narayanan, P.: Geometric structure computation from conics. In: Proc. Indian Conference on Computer Vision, Graphics and Image Processing (ICVGIP). pp. 9–14. Citeseer (2004)

[31] Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular SLAM system. IEEE Transactions on Robotics **31**(5), 1147–1163 (2015)

[32] Navab, N., Appel, M.: Canonical representation and multi-view geometry of cylinders. International Journal of Computer Vision **70**(2), 133–149 (2006)

[33] Oskarsson, M., Åström, K.: Accurate and automatic surveying of beacon positions for a laser guided vehicle. In: Proc. European Consortium for Mathematics in Industry (ECMI) (1998)

[34] Oskarsson, M., Åström, K., Overgaard, N.C.: Minimal cases of the structure and motion problem with missing data for one-dimensional retinae. In: Proc. Scandinavian Conference on Image Analysis (SCIA). pp. 482–489. Springer (2001)

[35] Oskarsson, M., Åström, K., Overgaard, N.C.: The minimal structure and motion problems with missing data for 1d retina vision. Journal of Mathematical Imaging and Vision **26**(3), 327–343 (2006)

[36] Pautrat, R., Lin, J.T., Larsson, V., Oswald, M.R., Pollefeys, M.: SOLD2: Self-supervised occlusion-aware line description and detection. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 11368–11378. IEEE (2021)

[37] Quan, L.: Invariant of a pair of non-coplanar conics in space: Definition, geometric interpretation and computation. In: Proc. International Conference on Computer Vision. pp. 926–931. IEEE (1995)

[38] Quan, L.: Conic reconstruction and correspondence from two views. IEEE Transactions on Pattern Analysis and Machine Intelligence **18**(2), 151–160 (1996)

[39] Quan, L.: Uncalibrated 1D projective camera and 3D affine reconstruction of lines. In: Proc. Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). pp. 60 – 65. IEEE (1997)

[40] Raposo, C., Barreto, J.P.: Theory and practice of structure-from-motion using affine correspondences. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5470–5478. IEEE (2016)

[41] Rublee, E., Rabaud, V., Konolige, K., Bradski, G.: ORB: An efficient alternative to SIFT or SURF. In: Proc. International conference on computer vision (ICCV). pp. 2564–2571. IEEE (2011)

[42] Sattler, T., Maddern, W., Toft, C., Torii, A., Hammarstrand, L., Stenborg, E., Safari, D., Okutomi, M., Pollefeys, M., Sivic, J., et al.: Benchmarking 6dof outdoor visual localization in changing conditions. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 8601–8610. IEEE (2018)

[43] Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4104–4113. IEEE (2016)

[44] Schönberger, J.L., Zheng, E., Pollefeys, M., Frahm, J.M.: Pixelwise view selection for unstructured multi-view stereo. In: European Conference on Computer Vision (ECCV). pp. 501–518. Springer (2016)

[45] Stewénius, H.: Gröbner Basis Methods for Minimal Problems in Computer Vision. Ph.D. thesis, Lund University (2005)

[46] Triggs, B., McLauchlan, P.F., Hartley, R.I., Fitzgibbon, A.W.: Bundle adjustment: A modern synthesis. In: Vision Algorithms: Theory and Practice. LNCS, Springer (2000)

[47] Xu, Y., Xu, W., Cheung, D., Tu, Z.: Line segment detection using transformers without edges. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4257–4266. IEEE (2021)

[48] Xue, N., Bai, S., Wang, F.D., Xia, G.S., Wu, T., Zhang, L., Torr, P.H.: Learning regional attraction for line segment detection. IEEE transactions on pattern analysis and machine intelligence **43**(6), 1998–2013 (2019)

[49] Xue, N., Wu, T., Bai, S., Wang, F., Xia, G.S., Zhang, L., Torr, P.H.: Holistically-attracted wireframe parsing. In: Proc. Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2788–2797. IEEE (2020)

[50] Zheng, J., Zhang, J., Li, J., Tang, R., Gao, S., Zhou, Z.: Structured3D: A large photo-realistic dataset for structured 3D modeling. In: Proceedings of The European Conference on Computer Vision (ECCV). pp. 519–535. Springer (2020)

[51] Zhou, Y., Qi, H., Ma, Y.: End-to-end wireframe parsing. In: Proc. International Conference on Computer Vision. pp. 962–971. IEEE (2019)

Paper V

Efficient Merging of Maps and Detection of Changes

Gabrielle Flood, David Gillsjö, Anders Heyden and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: With the advent of cheap sensors and computing capabilities as well as better algorithms it is now possible to do structure from motion using crowd sourced data. Individual estimates of a map can be obtained using structure from motion (SfM) or simultaneous localization and mapping (SLAM) using e.g. images, sound or radio. However the problem of map merging as used for collaborative SLAM needs further attention. In this paper we study the basic principles behind map merging and collaborative SLAM. We develop a method for merging maps – based on a small memory footprint representation of individual maps – in a way that is computationally efficient. We also demonstrate how the same framework can be used to detect changes in the map. This makes it possible to remove inconsistent parts before merging the maps. The methods are tested on both simulated and real data, using both sensor data from radio sensors and from cameras.

Keywords: Map Merging, Change Detection, Collaborative SLAM, Structure from Motion

1 Introduction

Structure from motion [5], is the problem of estimating the parameters of a map and of sensor motion using only sensor data. The map is typically a set of 2D or 3D points each consisting of a position and a feature vector. Assuming that feature errors are zero-mean Gaussian, the maximum likelihood estimate is that of minimising the sum of squares of the residuals. Within the field of computer vision this process is denoted bundle adjustment, where *bundle* refers to the bundle of light rays connecting each camera with each 3D point. For an overview of the literature and theory, see [13].

These optimization techniques are applicable not only to vision, but also to other types of sensors, such as audio, [9, 14] and radio [1]. With the advent of cheaper sensors and computing capabilities as well as better algorithms, it is now possible to gather and use much larger datasets. Instead of mapping a city every 5 years using special measurement cars or aerial photography, it is in principle possible for every car to add to the map of cities as they drive through them. Thus there is an additional need for research on map

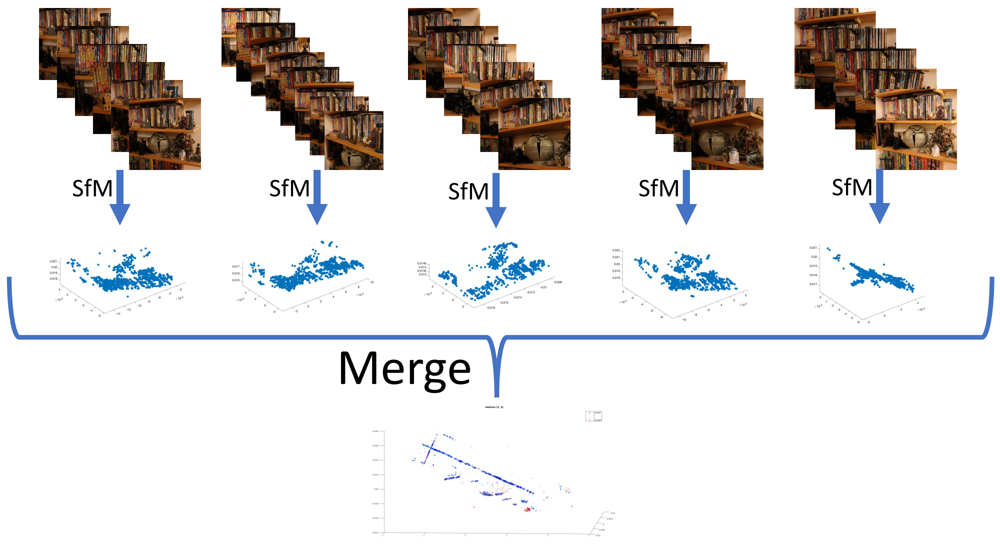


Figure 1: Structure from motion (SfM) is used to estimate a 3D map of scene features using images (or other sensors). In this paper we study the problem of detecting changes and merging maps, given multiple maps estimated by SfM from datasets collected at different occasions.

merging, including the problem of determining what has changed in a map. In this paper we study the basic principles behind map merging and collaborative SLAM. A straightforward method to merge several individual maps is to take all measurements into account simultaneously. However, non-linear optimization using all data can be prohibitively slow. We will study how a small memory footprint representation of a map can be generated and used to merge maps in a way that is computationally efficient, while still retaining most of the information from each individual bundle adjustment. We also demonstrate how the same framework can be used to detect changes in the map. This makes it possible to remove changing parts before merging the stationary parts of the map. The idea is demonstrated in Figure 1.

The idea of approximating the result from parts of the data has previously been used in the rotation averaging literature, cf. [2]. These approximate methods can give satisfactory results at a much increased speed. Another example of this idea is the approach of Global Epipolar Adjustment [12], in which a simplified error metric is based on the linear epipolar constraints for image pairs. Another approach is incremental light bundle adjustment, iLBA, [6] in which an error metric based on a combination of epipolar constraints and a variant of the trifocal constraint is used.

The main contributions of this paper are a novel method for computationally efficient merging of individual maps obtained from bundle adjustment, utilizing a compact representation of the Jacobian matrix, and a change detection method based on a statistical analysis of the residuals.

2 The Separate Bundles - for TOA and Images

Before different maps are merged, the individual map estimates have to be created. In this section we present some of the notations used to understand how the raw data relates to the quality of the map estimates.

For the case of time of arrival (TOA) measurements the feature map consists of a number of receiver positions. Initially, TOA measures between m receivers at positions $x_i \in \mathcal{R}^3$ and n sender positions $y_j \in \mathcal{R}^3$ are given. For each sender-receiver pair this measure can be translated into a distance estimate $d_{ij} = |x_i - y_j| + \varepsilon_{ij}$, where $1 \leq i \leq m$ and $1 \leq j \leq n$ and where $|\cdot|$ denotes the Euclidean norm of a vector in $\mathcal{R}^3$. The measurements errors ε_{ij} are assumed to be independent, Gaussian with mean zero and standard deviation σ.

The final map estimate for a TOA or structure from motion system is usually obtained by non-linear least squares minimization over inlier measurements; this process is referred to as bundle adjustment in computer vision. Here, a few key components from the optimization are presented.

For the TOA data, let $\mathbf{r}$ denote the measurements residuals,

$$\mathbf{r} = \begin{bmatrix} r_{11} & \dots & r_{1n} & r_{21} & \dots & r_{2n} & r_{m1} & \dots r_{mn} \end{bmatrix}^T, \qquad r_{ij} = d_{ij} - |x_i - y_j|, \quad (1)$$

and denote the parameters of interest, which are optimized, by $\mathbf{z}$. This would typically be the receiver and the sender positions,

$$\mathbf{z} = (x_1, x_2, \dots x_m, y_1, \dots y_n). \quad (2)$$

The computer vision case is analogous. Denoting the camera matrices P_i and the 3D points U_j, each image point u_{ij} gives a residual r_{ij}. The residual vector $\mathbf{r}$ is found by stacking all image feature residuals r_{ij} and the parameters are collected in a parameter vector

$$\mathbf{z} = (P_1, P_2, \dots P_m, U_1, \dots U_n). \quad (3)$$

The maximum likelihood estimate of $\mathbf{z}$ is found by minimizing the sum of the squares of the residuals, i.e.

$$\mathbf{z}^* = \text{argmin}_{\mathbf{z}} \mathbf{r}^T \mathbf{r}, \quad (4)$$

which gives the optimal parameter update

$$\Delta \mathbf{z} = -(J^T J)^{-1} J^T \mathbf{r}. \quad (5)$$

For more details on the optimization, see [13]. For the analysis, the estimate of the matrix J (the Jacobian) is containing the derivatives of the residuals with respect to the parameters is of interest, i.e. $\mathbf{r}$ with respect to $\mathbf{z}$, further on denoted $\partial \mathbf{r} / \partial \mathbf{z}$.

The map points can only be estimated up to a choice of coordinate system. For simplicity we will in the TOA case normalize the coordinate system so that the first receiver is placed in the origin, the second along the x-axis, the third in the xy-plane and so forth. By removing this gauge freedom with dimension ϕ we see that the effective number of degrees of freedom in the problem is $d_{dof} = (m+n)\rho - \phi$, where ρ denotes the dimension. For TOA problems in 3D we have $\rho = 3$ and $\phi = 6$. The effective degrees of freedom for the computer vision case becomes $d_{dof} = (6m+3n) - \phi$, with gauge freedom $\phi = 7$ since we are free to choose position, orientation and scale of the coordinate system.

3 Merging Separate Maps

Once the N separate maps are obtained they can be merged to get a single more accurate map. We have investigated three different ways to do this.

3.1 The Full Bundle

One way to add the maps is do one large bundle where all the individual measurements are used simultaneously. Merging all maps through a large bundle is a good way to get an accurate map. However, the method is time consuming and if a new measurement is made after the original merge, the whole map has to be re-bundled. In that sense, there is no way to *add* new information to the existing, which makes this method unsuitable for online applications.

3.2 The Kalman Filter

A traditional method designed to update parameters gradually is the Kalman filter [8]. The algorithm for the Kalman filter looks as follows:

Priori estimate update:

$$x_1 = A \cdot x_0 \tag{6}$$

$$P_1 = A \cdot P_0 \cdot A^T + Q \tag{7}$$

Measurement update:

$$K = P_1 \cdot H^T \cdot (H \cdot P_1 \cdot H^T + R)^{-1} \tag{8}$$

$$x_2 = x_1 + K \cdot (u - H \cdot x_1) \tag{9}$$

$$P_2 = (I - K \cdot H) \cdot P_1 \, . \tag{10}$$

Then, $H \cdot x_2$ is the new state prediction, and x_2 and P_2 are the new estimates replacing x_0 and P_0 for the next iteration. In our case x_0 will be the receivers from the first measurement occasion, $x_0 = \mathbf{q}^{(1)}$ (superscript denoting measurement occasion), while the observation u will be the receiver values from the following $N-1$ measurements s.t. $u_{k-1} = \mathbf{q}^{(k)}$, $2 \leq k \leq N$. Both the update matrix and the observation matrix are identity matrices,

$A = I$, $H = I$ and the covariance of the random excitation is set to $Q = 0.1 \cdot I$. Finally, P_0 and R are measurement uncertainties, $P_0 = \mathbf{C}[\Delta\mathbf{q}^{(1)}]$ and $R_{k-1} = \mathbf{C}[\Delta\mathbf{q}^{(k)}]$, $2 \leq k \leq N$. The covariance $\mathbf{C}[\Delta\mathbf{q}]$ can be extracted from the covariance of $\Delta\mathbf{z}$ from Equation (5). This is given by

$$\mathbf{C}[\Delta\mathbf{z}] = (J^T J)^{-1} J^T \cdot \mathbf{E}[\mathbf{r}^T\mathbf{r}] \cdot J(J^T J)^{-1} = \sigma^2 (J^T J)^{-1} \,. \tag{11}$$

The covariance of the map, $\mathbf{C}[\Delta\mathbf{q}]$, can be retrieved by picking the rows and columns in $\mathbf{C}[\Delta\mathbf{z}]$ that correspond to $\mathbf{q}$ and the variance of $\mathbf{r}$ can be approximated by [7, p. 148]

$$\sigma^2 \approx \frac{1}{m \cdot n - d_{dof}} \cdot \mathbf{r}^T\mathbf{r} = \frac{1}{m \cdot n - d_{dof}} \cdot \sum_{i=1}^{m \cdot n} r_i^2 \,. \tag{12}$$

The Kalman filter is a computationally cheap method. However, it is not as accurate as the full bundle. Also, the parameters need to be tuned for the specific problem and it is not evident either how to detect and handle changes in the map.

3.3 The Linearized Method

The idea of this method is that the optimal residuals from the separate bundles can be linearized – such that all that needs to be saved is a small memory footprint representation – to avoid the large bundles. Having the optimal residuals $\mathbf{r}^{(k)}$ and the optimal Jacobians $J^{(k)}$ from each run k, the residuals can be linearized using a first order Taylor approximation. A key idea here is to divide the unknown parameters in $\mathbf{z}$ into two parts $\mathbf{q}$ and $\mathbf{s}$, where $\mathbf{q}$ are the parameters that exist in several SLAM sessions. The parameters $\mathbf{s}$ can be thought of as auxillary paramters, e.g. those that are relevant only for one specific bundle session. In the time-of-arrival case, some of the 3D anchors might be constant over several SLAM sessions whereas the measurement points and some of the anchors might be different. For vision based structure from motion, some of the 3D points are the same (these go into $\mathbf{q}$) whereas the rest of the points and camera matrices go into $\mathbf{s}$.

The Compressed Residual

First, the Jacobian is divided into two blocks

$$J = \begin{bmatrix} J_a & J_b \end{bmatrix} \,, \tag{13}$$

where J_a contains the columns that correspond to the main parameters $\mathbf{q}$ and J_b contains the columns corresponding to the auxiliary parameters $\mathbf{s}$. The squared Jacobian is

$$J^T J = \begin{bmatrix} J_a^T \\ J_b^T \end{bmatrix} \cdot \begin{bmatrix} J_a & J_b \end{bmatrix} = \begin{bmatrix} J_a^T J_a & J_a^T J_b \\ J_b^T J_a & J_b^T J_b \end{bmatrix} = \begin{bmatrix} U & W \\ W^T & V \end{bmatrix} \,. \tag{14}$$

Furthermore, if we insert this in the equation for the optimal update from (5) we get

$$\Delta \mathbf{z} = \begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{s} \end{bmatrix} = -(J^T J)^{-1} J^T \mathbf{r} \qquad \Leftrightarrow \qquad \begin{bmatrix} U & W \\ W^T & V \end{bmatrix} \begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{s} \end{bmatrix} = -J^T \mathbf{r} \,. \tag{15}$$

The product $-J^T \mathbf{r}$ is zero in an optimal point and so the second row provides a connection between $\mathbf{q}$ and $\mathbf{s}$. This gives a linear constraint on how to adjust the auxiliary parameters $\mathbf{s}$ when the main parameters $\mathbf{q}$ change. Thus the partial derivatives of $\mathbf{s}$ with respect to $\mathbf{q}$ is

$$W^T \Delta \mathbf{q} + V \Delta \mathbf{s} = 0 \quad \Leftrightarrow \quad \Delta \mathbf{s} = -V^{-1} W^T \Delta \mathbf{q} \quad \Rightarrow \quad \frac{\partial \mathbf{s}}{\partial \mathbf{q}} = -V^{-1} W^T \,. \tag{16}$$

We can use this together with the definition $J = \partial \mathbf{r} / \partial \mathbf{z}$ to find how the residuals change if we change the receiver map

$$\Delta \mathbf{r} = \begin{bmatrix} J_a & J_b \end{bmatrix} \begin{bmatrix} \Delta \mathbf{q} \\ \Delta \mathbf{s} \end{bmatrix} = \left(J_a + J_b \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{q}} \right) \Delta \mathbf{q} \,. \tag{17}$$

Thus, $J_a + J_b \frac{\partial \mathbf{s}}{\partial \mathbf{q}}$ will be the Jacobian for the map, further on denoted J_q.

Now, denote the residuals as a function of $\Delta \mathbf{q}$. A first order Taylor expansion gives

$$\mathbf{r}(\Delta \mathbf{q}) \approx \mathbf{r}|_o + \mathbf{r}'_{\Delta \mathbf{q}}|_o \Delta \mathbf{q} = \mathbf{r}|_o + J_q|_o \Delta \mathbf{q} \,. \tag{18}$$

Here o denotes an optimal point and $|_o$ denotes evaluating an expression at the point o. Then, the square of these residuals will be

$$\mathbf{r}^T \mathbf{r} \approx (\mathbf{r}|_o + J_q|_o \Delta \mathbf{q})^T (\mathbf{r}_o + J_q|_o \Delta \mathbf{q}) = \mathbf{r}|_o^T \mathbf{r}_o + 2\mathbf{r}|_o^T J_q|_o \Delta \mathbf{q} + \Delta \mathbf{q}^T J_q|_o^T J_q|_o \Delta \mathbf{q} \,. \tag{19}$$

In a minimum point $\mathbf{r}|_o^T J_q$ is zero. Furthermore, using the QR-decomposition of the Jacobian we get

$$\Delta \mathbf{q}^T J_q^T J_q \Delta \mathbf{q} = \Delta \mathbf{q}^T (QR)^T QR \Delta \mathbf{q} = \Delta \mathbf{q}^T R^T Q^T QR \Delta \mathbf{q} = \Delta \mathbf{q}^T R^T R \Delta \mathbf{q} \,. \tag{20}$$

Introducing the notation $a = (\mathbf{r}|_o^T \mathbf{r}|_o)^{1/2}$, the squared residuals from (19) can be written shorter as

$$\mathbf{r}^T \mathbf{r} \approx a^2 + \Delta \mathbf{q}^T R^T R \Delta \mathbf{q} \,, \tag{21}$$

and this is our compressed expression for the squared residuals.

The Merge

Furthermore, this compressed expression can be used to add two separate maps. Assume that we have the residuals for the two maps,

$$\left(\mathbf{r}^{(i)}\right)^T \left(\mathbf{r}^{(i)}\right) = \left(a^{(i)}\right)^2 + \left(\Delta \mathbf{q}^{(i)}\right)^T \left(R^{(i)}\right)^T R^{(i)} \Delta \mathbf{q}^{(i)} \,, \qquad i = 1, 2 \,. \tag{22}$$

Adding the two equations and writing $\Delta\mathbf{q}^{(i)} = \mathbf{q} - \mathbf{q}^{(i)}$ for an arbitrary $\mathbf{q}$ gives

$$\begin{aligned}\sum_{i=1}^{2} \left(\mathbf{r}^{(i)}\right)^T \left(\mathbf{r}^{(i)}\right) &= \sum_{i=1}^{2} \left(a^{(i)}\right)^2 + \left(\Delta\mathbf{q}^{(i)}\right)^T \left(R^{(i)}\right)^T R^{(i)} \Delta\mathbf{q}^{(i)} = \left(a^{(1)}\right)^2 + \left(a^{(2)}\right)^2 \\ &+ \begin{bmatrix} R^{(1)}\left(\mathbf{q} - \mathbf{q}^{(1)}\right) \\ R^{(2)}\left(\mathbf{q} - \mathbf{q}^{(2)}\right) \end{bmatrix}^T \begin{bmatrix} R^{(1)}\left(\mathbf{q} - \mathbf{q}^{(1)}\right) \\ R^{(2)}\left(\mathbf{q} - \mathbf{q}^{(2)}\right) \end{bmatrix} = \left(a^{(1)}\right)^2 + \left(a^{(2)}\right)^2 + \hat{\mathbf{r}}^T \hat{\mathbf{r}}\,. \end{aligned} \tag{23}$$

The terms $(a^{(1)})^2$ and $(a^{(2)})^2$ are fixed while the third term $\hat{\mathbf{r}}^T\hat{\mathbf{r}}$ can be minimized to minimize the sum of the residuals. Introducing new notations M and b, $\hat{\mathbf{r}}$ can be written

$$\hat{\mathbf{r}} = \begin{bmatrix} R^{(1)}(\mathbf{q} - \mathbf{q}^{(1)}) \\ R^{(2)}(\mathbf{q} - \mathbf{q}^{(2)}) \end{bmatrix} = \begin{bmatrix} R^{(1)} \\ R^{(2)} \end{bmatrix} \mathbf{q} - \begin{bmatrix} R^{(1)}\mathbf{q}^{(1)} \\ R^{(2)}\mathbf{q}^{(2)} \end{bmatrix} = M\mathbf{q} - b\,. \tag{24}$$

To minimize $\hat{\mathbf{r}}$ and thus $\hat{\mathbf{r}}^T\hat{\mathbf{r}}$ is a least squares problem which can be solved using the pseudo inverse. Denoting the merged map $\mathbf{q}^*$ gives

$$\mathbf{q}^{(*)} = (M^T M)^{-1} M^T b\,. \tag{25}$$

We can also compress the final result. Using that a general $\mathbf{q}$ can be written $\mathbf{q} = \Delta\mathbf{q}^{(*)} + \mathbf{q}^{(*)}$, the third term in (23) can be expressed

$$\begin{aligned}\hat{\mathbf{r}}^T\hat{\mathbf{r}} =& (M\mathbf{q} - b)^T (M\mathbf{q} - b) = \left(M\mathbf{q}^{(*)} - b + M\Delta\mathbf{q}^{(*)}\right)^T \left(M\mathbf{q}^{(*)} - b + M\Delta\mathbf{q}^{(*)}\right) \\ =& \left(M\mathbf{q}^{(*)} - b\right)^T \left(M\mathbf{q}^{(*)} - b\right) + \left(\Delta\mathbf{q}^{(*)}\right)^T M^T M \Delta\mathbf{q}^{(*)}\,, \end{aligned} \tag{26}$$

where the linear term vanishes due to orthogonality. Using this in Equation (23) gives

$$\left(\mathbf{r}^{(*)}\right)^T \mathbf{r}^{(*)} = \left(a^{(1)}\right)^2 + \left(a^{(2)}\right)^2 + \left(M\mathbf{q}^{(*)} - b\right)^T \left(M\mathbf{q}^{(*)} - b\right) + \left(\Delta\mathbf{q}^{(*)}\right)^T M^T M \Delta\mathbf{q}^{(*)}\,. \tag{27}$$

If M is QR-decomposed in a similar manner as J_q was in (20) this total result can be compressed as

$$\left(\mathbf{r}^{(*)}\right)^T \mathbf{r}^{(*)} = \left(a^{(*)}\right)^2 + \left(\Delta\mathbf{q}^{(*)}\right)^T \left(R^{(*)}\right)^T R^{(*)} \Delta\mathbf{q}^{(*)}\,, \tag{28}$$

with $R^{(*)}$ being the triangular matrix from the QR-decomposition of M and

$$a^{(*)} = \left(\left(a^{(1)}\right)^2 + \left(a^{(2)}\right)^2 + \left(M\mathbf{q}^{(*)} - b\right)^T \left(M\mathbf{q}^{(*)} - b\right)\right)^{\frac{1}{2}}\,. \tag{29}$$

By this, the representation of the final map is the same as in (21) and the merged map can be treated as one of the original. Furthermore, more maps can be added using the algorithm described above. Thus, to add maps, all we need to save from the separate bundles are the maps $\mathbf{q}^{(i)}$, the squared residuals $a^{(i)}$, and the triangular matrices $R^{(i)}$ from the QR-decompositions of the Jacobians.

In some cases the linearized method is similar to the Kalman filter. However, several maps can be added at once using the linearized model and it also allows for better control. We will also show that this method can be developed to detect map changes.

4 Detection of Changes

Once we know how to merge two or more maps we can also use this to detect whether the map has changed between the measurement occasions. For this, assume that we have two maps $\mathbf{q}^{(1)}$ and $\mathbf{q}^{(2)}$ and their merge $\mathbf{q}^{(*)}$. Furthermore, we have the norms of their residuals, $a^{(1)}$, $a^{(2)}$ and $a^{(*)}$. An approximation for the residual variance is derived in (12). This can be used to find the estimated value of how the squared residuals change when we add maps. Rearranging terms from (12), we get

$$\mathbf{E}\left[\left(a^{(i)}\right)^2\right] = \mathbf{E}\left[\left(\mathbf{r}^{(i)}\right)^T\left(\mathbf{r}^{(i)}\right)\right] = \sigma^2(mn - (n\rho + m\rho - \phi))\,, \qquad i = 1, 2 \tag{30}$$

$$\mathbf{E}\left[\left(a^{(*)}\right)^2\right] = \mathbf{E}\left[\left(\mathbf{r}^{(*)}\right)^T\left(\mathbf{r}^{(*)}\right)\right] = \sigma^2(mn - (Nn\rho + m\rho - \phi))\,, \tag{31}$$

and subtracting these – in this case with $N = 2$ maps – gives

$$\mathbf{E}\left[\left(a^{(*)}\right)^2 - \left(a^{(1)}\right)^2 - \left(a^{(2)}\right)^2\right] = \sigma^2(N-1)(m\rho - \phi)\,. \tag{32}$$

If we use real data, σ is unknown, but it can be estimated from the separate bundles using (12), s.t. $\hat{\sigma}^2 = ((\sigma^{(1)})^2 + (\sigma^{(2)})^2)/2$.

The values in (32) can be seen as a sum of $(N-1)(m\rho - \phi)$ Gaussian variables, and a sum of 2ν independent Gaussian distributed variables with mean zero and standard deviation σ_n has a Γ distribution with density [3, p. 47]

$$f_{\alpha,\nu}(x) = \frac{1}{\Gamma(\nu)}\alpha^\nu x^{\nu-1}e^{-\alpha x}\,, \tag{33}$$

with $\alpha = 1/(2\sigma_n^2)$ and Γ being the gamma function. This density will be denoted $\Gamma(\alpha, \nu)$ (two parameters). Furthermore, using $\tilde{a} = (a^{(1)})^2 + (a^{(2)})^2 - (a^{(*)})^2$ and $\gamma = (N-1)(m\rho - \phi)$ we get that $\tilde{a} \sim \Gamma(1/(2\sigma^2), \gamma/2)$. Thus, to know whether a map has changed we can compare the estimated $\tilde{a}$ to the distribution. A reasonable choice is that if the difference $\tilde{a}$ lies within the 99 percentile of $\Gamma(1/(2\sigma^2), \gamma/2)$ there has not been any change in the map, but if $\tilde{a}$ is higher than this limit, a change has probably occured.

If a change between two maps is discovered, we further investigate those maps. By comparing the positions for each map point, we say that if the distance between them is larger than $3\hat{\sigma}$ the map point has probably moved. This could also be used to decrease the variance even further for the receivers that have not changed, by using information from all maps for these receivers.

5 Experimental Validation

To validate the method suggested in this paper, experiments on simulated TOA data as well as real ultra-wideband (UWB) data have been performed. We have also developed the

Table 1: The results from the experiment explained in Section 5.1. The values come from a merge of two maps between which no change has occured. These values are the mean of 10 similar runs.

	n	10	100	1000	4000
Runtime [s]	Full bundle	$2.3 \cdot 10^{-2}$	0.19	3.8	54.7
	Linearized	$1.9 \cdot 10^{-3}$	$3.2 \cdot 10^{-4}$	$4.7 \cdot 10^{-4}$	$3.3 \cdot 10^{-4}$
	Kalman	$2.4 \cdot 10^{-3}$	$2.1 \cdot 10^{-4}$	$2.2 \cdot 10^{-4}$	$2.1 \cdot 10^{-4}$
$\|\| \|\mathbf{q}^{(t)} - \mathbf{q}\| \|\|$	Full bundle	1.20	0.11	$1.6 \cdot 10^{-2}$	$3.0 \cdot 10^{-3}$
	Linearized	1.34	0.11	$1.6 \cdot 10^{-2}$	$3.0 \cdot 10^{-3}$
	Kalman	1.48	0.12	$2.2 \cdot 10^{-2}$	$5.8 \cdot 10^{-3}$
$\frac{\mathbf{r}^T\mathbf{r}}{mn} = \frac{a^2}{mn}$	Full bundle	0.11	0.12	0.13	0.13
	Linearized	0.11	0.12	0.13	0.13

method to work for, and tried it on, 3D-reconstructions from image data.

5.1 Time of Arrival – Simulated Data

For each of the simulated experiments m receivers in 3D were generated from a uniform distribution, $\mathbf{q}^{(t)} \sim \mathcal{U}(0, 10)$, superscript (t) denoting the true value. We simulated N different measurement occasions with n sender positions $\mathbf{s}^{(t)} \sim \mathcal{U}(0, 10)$ each and calculated the mn sender-receiver distances. Gaussian noise with standard deviation σ was added to achieve distance measurements. For each measure we performed a separate bundle to get the N maps $\mathbf{q}^{(1)}, \dots, \mathbf{q}^{(N)}$ and the compressed representation explained in Section 3.3 and more specifically in (21).

Test of Time and Accuracy

For the first experiment $m = 10$, $\sigma_n = 0.3$, $N = 2$ and no change occured in the true map. The experiments were run four times with $n = 10, 100, 1000, 4000$ respectively. For each case, the merge was computed using the three methods presented in this paper and the runtimes were measured. We computed the error norm $\sqrt{\sum_{i=1}^{m} |q_i^{(t)} - q_i|^2}$ and for the full bundle and the linearized method, we also computed the squared distance residuals per residual $\mathbf{r}^T\mathbf{r}/(mn) = a^2/(mn)$. The results can be seen in Table 1.

Even if the runtime is highly dependent on the implementations, the table gives a valid comparison between the methods. The linearized method is almost as accurate as the full bundle. Moreover, when only the sender positions increase, and thus also the number of distances, the runtime for the linearized method and the Kalman filter do not increase notably, while the runtime for the full bundle does. Hence, the linearized method is faster than the full bundle and more accurate than the Kalman filter.

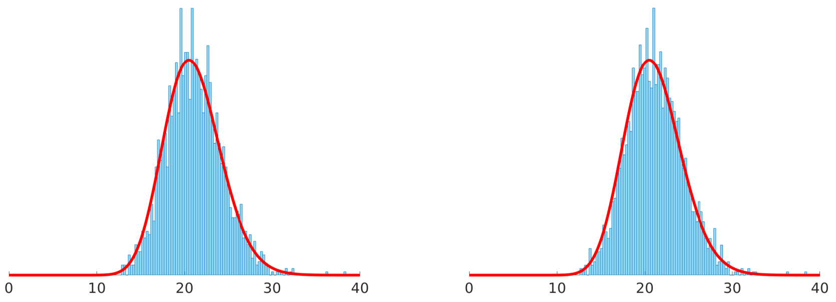


Figure 2: The plots show histograms of the residuals $\tilde{a}_{full}$ (to the left) and $\tilde{a}_{lin}$ (to the right) computed using the full bundle and the linearized method respectively. The curve (–) shows the Γ distribution which we expect $\tilde{a}$ to belong to.

Validating the Detection Threshold

To validate the threshold for detection of changes described in Section 4, we tested the distribution of $\tilde{a}$ empirically. Using $m = 30$, $n = 200$, $N = 2$ and $\sigma_n = 0.5$ the distances were computed. The separate bundles as well as the merge using both the full bundle and the linearized method were then conducted. For all of the different maps we computed the compressed representations from (21). We then computed

$$\tilde{a}_{full} = \left(a^{(1)}\right)^2 + \left(a^{(2)}\right)^2 - \left(a^{(*)}_{full}\right)^2, \quad \text{and} \quad \tilde{a}_{lin} = \left(a^{(1)}\right)^2 + \left(a^{(2)}\right)^2 - \left(a^{(*)}_{lin}\right)^2, \quad (34)$$

where subscript index *full* and *lin* denotes the full bundle and the linearized method respectively. This was re-made 2000 times with different noise. The total degrees of freedom were $\gamma = (N-1)(m \cdot \rho - \phi) = 30 \cdot 3 - 6 = 84$. The results of $\tilde{a}_{full}$ and $\tilde{a}_{lin}$ were then plotted in a histogram together with a $\Gamma(2, 42)$ distribution in Figure 2. The histograms agree well with the gamma distribution in both cases; hence, this can be used to test the significance.

Detection of Changed Maps

Furthermore, we did an experiment where the map actually had changed. This time we used $m = 10$, $n = 30$, $N = 3$ and $\sigma_n = 0.5$. Four of the ten receivers moved before the last measurement. After running the separate bundles and merging the maps both using a full bundle and our linearized method we investigated the differences in the residuals. The system had $\gamma = 2 \cdot (10 \cdot 3 - 6) = 48$ degrees of freedom and thus $\tilde{a}$ should be such that it could come from a $\Gamma(1/(2\hat{\sigma}^2), 24)$ distribution if no changes has occured. Using the estimated $\hat{\sigma}^2$ the 99-percentile of this was $\tilde{a} = 17.7$. In this specific case, the results from the merge gave $\tilde{a}_{full} = 603$ and $\tilde{a}_{lin} = 749$ and this clearly showed that something had changed. The results from the unsuccessful merge can be seen to the left in Figure 3. To

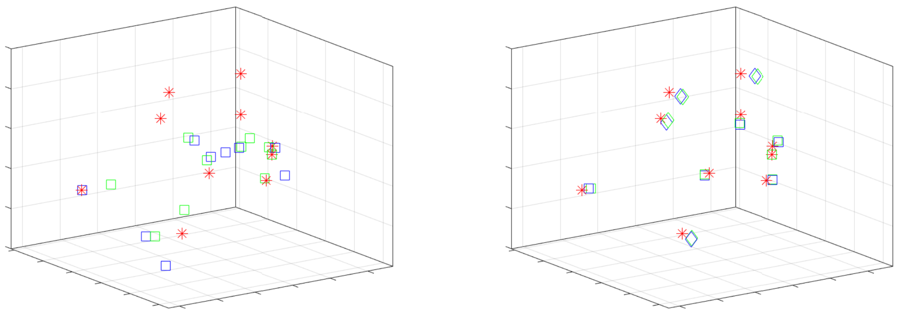


Figure 3: An unsuccessful merge of map 1,2 and 3 (left) and a successful merge of map 1 and 2 (right). The stars ($*$) show the true receiver positions, the squares the results from full bundle (□) and the linearized method (□). In the right figure, the points for which a change has been detected are (correctly) marked by a diamond (◇,◇).

the right in Figure 3 are the results from the merge between the first and second map, after the system successfully had detected the change.

5.2 Time of Arrival – Real Data

To test our method $N = 9$ experiments were conducted using a Bitcraze Crazyflie quadcopter and their Loco-positioning system which consists of $m = 5$ anchors with UWB chips and a flying quadcopter with a mounted UWB chip, giving approximately $n = 600$ sender positions for each measurement. The five anchors were positioned around the room and one of them was moved before the last three runs. The experiment was conducted in a MOCAP studio to record the ground truth flightpath as well as the anchor positions. Distance measurements from the quadcopter (sender) to all the anchors (receivers) were measured at a frequency of 30 Hz.

The problem was solved as explained in previous sections, except that the threshold for $\tilde{a}$ now was 10 times the 99 percentile for the Γ distribution. This threshold was used for all real data experiments. In Figure 4 the results from the Kalman filter and the linearized method are shown. While the dynamics of the Kalman filter makes the estimated receivers end up further away from the true positions – *on their way* to the correct position – for some of the measurements, the linearized method correctly detects when a change has occured. Thereafter, only the similar maps are merged.

5.3 Images – Real Data

In this experiment, $N = 5$ sets of images were taken of an indoor scene, a bookshelf with a number of toy models, as depicted in Figure 1. In between set 2 and 3 an R2D2 model

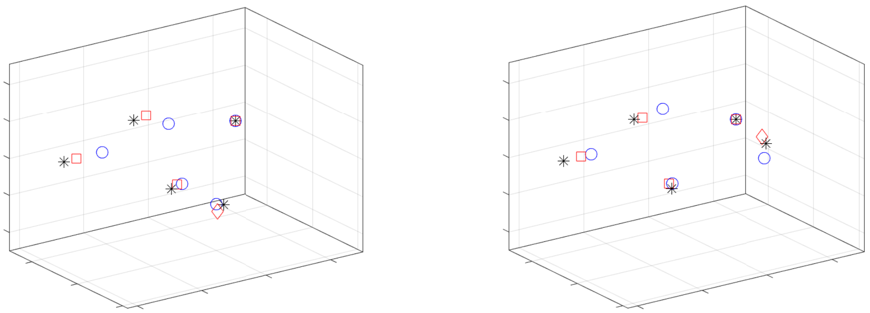


Figure 4: Results from two of the maps from the experiments with UWB data. The stars ($*$) show the true receiver positions, the circles ($\circ$) the results from the Kalman filter and the squares ($\square$) from the linearized method. The change between the maps has been correctly detected by the linearized method and changed receivers are marked with a diamond ($\diamond$).

was moved, which we wanted to detect. As a first step we used a structure from motion pipeline [11] to obtain a 3D reconstruction for each set. The points in this reconstruction are the feature points in the map, corresponding to the receivers in the TOA experiments.

Unlike the TOA experiments, correspondence between 3D points in the different datasets are not given. Prior to merging, we performed data association by SIFT [10] feature matching and geometric alignment in a RANSAC [4] framework. After this the maps were also in the same coordinate system, which is required for the linearized method and speeds up the full bundling method.

Using the same method as in Section 5.2 – with detection based on a Γ distribution and the feature point distances – the algorithm detected change during the merge of dataset 2 and 3, which is correct. In Figure 5 we see that the feature points on R2D2 are correctly detected as changed. Note that some features are not present in both datasets and therefore these features on the R2D2 are not marked as changed. Figure 6 shows the 3D reconstruction from above. Here we see that the merged points on R2D2 does not align with either dataset 2 or 3.

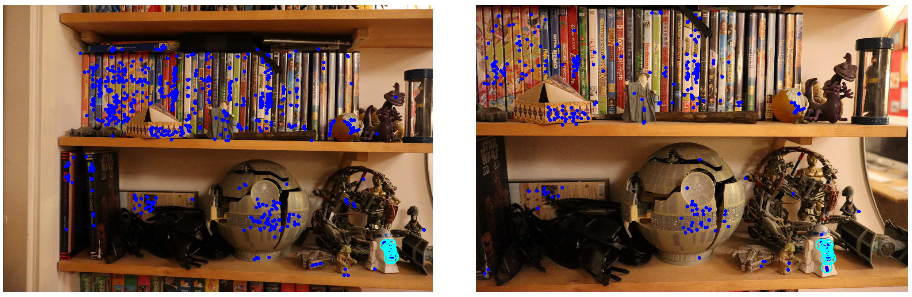


Figure 5: Changes detected in merge between dataset 2 and 3. Feature points are maked with blue dots and changed features are circled in cyan.

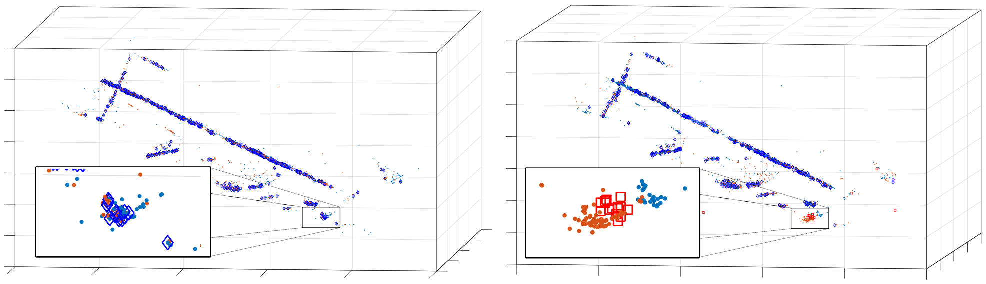


Figure 6: To the left, the merge between dataset 1 and 2 where no change was detected. The separate maps are marked with dots (●,●) and the merge by diamonds (◇). To the right, the merge between dataset 2 and 3, where a change was detected. The points for which a change was detected are marked by squares (□).

6 Conclusions

We have presented a novel and efficient method, with small memory footprint, for merging individual maps obtained from bundle adjustment optimization along with a statistically motivated method for detecting changes in the map. The method has been compared favorably to using full bundle adjustment and the Kalman filter and is shown to be a good compromise between performance and time efficiency. This makes the method suitable for online applications as well as the use of crowd sourced data. The performance has been confirmed on both TOA and vision problems for both simulated and real data. One limitation is that the map points used for the coordinate system normalization need to be consistent for all maps. However, if this problem is solved, we believe that the method could be further developed to a full collaborative SLAM system.

Acknowledgments

This work was partially supported by the strategic research projects ELLIIT and eSSENCE, the Swedish Foundation for Strategic Research project, Semantic Mapping and Visual Navigation for Smart Robots (grant no. RIT15-0038), and Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

References

[1] Batstone, K., Oskarsson, M., Åström, K.: Robust time-of-arrival self calibration and indoor localization using wi-fi round-trip time measurements. In: proc. of International Conference on Communication (2016)

[2] Enqvist, O., Olsson, C., Kahl, F.: Non-sequential structure from motion. In: Work-

shop on Omnidirectional Vision, Camera Networks and Non-Classical Cameras (OMNIVIS). Barcelona, Spain (2011)

[3] Feller, W.: An Introduction to Probability Theory and Its Applications, vol. 2. John Wiley & Sons (1968)

[4] Fischler, M.A., Bolles, R.C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM **24**(6), 381–395 (1981)

[5] Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision. Cambridge University Press (2004), second Edition.

[6] Indelman, V., Roberts, R., Beall, C., Dellaert, F.: Incremental light bundle adjustment. In: BMVC 2012 - Electronic Proceedings of the British Machine Vision Conference 2012 (01 2012)

[7] Jakobsson, A.: An Introduction to Time Series Modeling. Studentlitteratur AB, first edition edn. (2013)

[8] Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of basic Engineering **82**(1), 35–45 (1960)

[9] Kuang, Y., Burgess, S., Torstensson, A., Åström, K.: A complete characterization and solution to the microphone position self-calibration problem. In: ICASSP (2013)

[10] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (Nov 2004)

[11] Olsson, C., Enqvist, O.: Stable structure from motion for unordered image collections. In: Proceedings of 17th Scandinavian Conference, SCIA (2011)

[12] Rodríguez, A.L., López-de Teruel, P.E., Ruiz, A.: Reduced epipolar cost for accelerated incremental sfm. In: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on. pp. 3097–3104. IEEE (2011)

[13] Triggs, W., McLauchlan, P., Hartley, R., Fitzgibbon, A.: Bundle adjustment: A modern synthesis. In: Vision Algorithms: Theory and Practice, pp. 298–372. LNCS, Springer Verlag (2000)

[14] Zhayida, S., Andersson, F., Kuang, Y., Åström, K.: An automatic system for microphone self-localization using ambient sound. In: 22st European Signal Processing Conference (2014)

Paper VI

Generic Merging of Structure from Motion Maps with a Low Memory Footprint

Gabrielle Flood, David Gillsjö, Patrik Persson,
Anders Heyden and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: With the development of cheap image sensors, the amount of available image data have increased enormously, and the possibility of using crowdsourced collection methods has emerged. This calls for development of ways to handle all these data. In this paper, we present new tools that will enable efficient, flexible and robust map merging. Assuming that separate optimisations have been performed for the individual maps, we show how only relevant data can be stored in a low memory footprint representation. We use these representations to perform map merging so that the algorithm is invariant to the merging order and independent of the choice of coordinate system. The result is a robust algorithm that can be applied to several maps simultaneously. The result of a merge can also be represented with the same type of low-memory footprint format, which enables further merging and updating of the map in a hierarchical way. Furthermore, the method can perform loop closing and also detect changes in the scene between the capture of the different image sequences. Using both simulated and real data — from both a hand held mobile phone and from a drone — we verify the performance of the proposed method. **Keywords:** Structure from motion, Map Merging, Minimal Solvers

1 Introduction

Over the last couple of years the availability of cheap image sensors — such as cameras in mobile phones — has increased immensely. This allows for fast and relatively straightforward collection of large datasets through crowdsourcing. The images can be used to create 3D maps of the environment. However, the more data there are, the heavier the computations for creating these maps will be and due to this, there is a need for faster algorithms for creating 3D maps. Furthermore, additional research on how to fuse individual maps into one global, more accurate map is needed. One use case of such algorithms can be found in the industry for self-driving cars. With a fast and accurate way to merge individual submaps, each car that drives in an environment could create its own local map and use that to contribute to a global map.

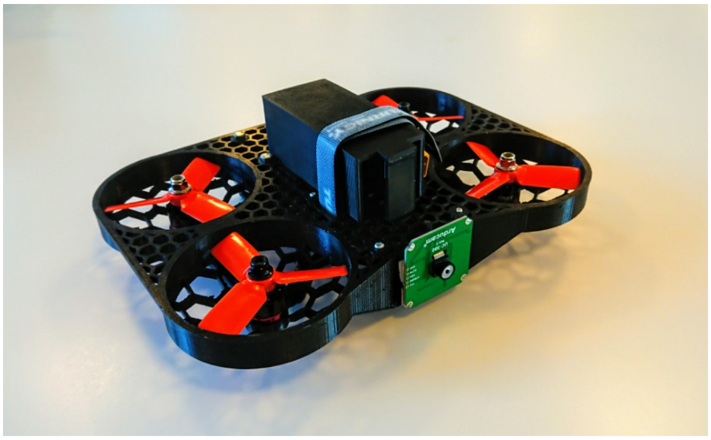


Figure 1: A drone equipped with a camera, IMU and a Raspberry Pi. The drone is used for real-time mapping and was used for one of the real data experiments in this paper.

Estimating map parameters and sensor motion using only sensor data is referred to as simultaneous location and mapping (SLAM) [1, 2] and structure from motion (SfM) [3]. Classically, SLAM has focused more on the motion, while SfM has been more focused on the structure. Also, SLAM often requires that one moving camera is used, while SfM can be used for unsorted images from different cameras. Nevertheless, the two methods essentially solve the same problem, but are originating from different research fields.

When image data are used, the SfM is usually performed using *bundle adjustment*. The name refers to the bundle of rays going from each 3D point in space to each camera and it can be seen as a large sparse geometric parameter estimation problem, cf. [4]. Bundle adjustment is commonly used both as a final step and as an intermediate step in the optimisation to prevent error buildup [5, 6]. It can also be used to merge maps, by doing a new optimisation over all data at once. However, bundle adjustment is a computationally expensive process and there is a need for making these methods more efficient.

A faster method to align two maps is to use point cloud registration. One example of a commonly used method for registration is iterative closest point (ICP) [7]. This does not require any knowledge of point matches between the different sets. If such matches are known, one can instead use e.g. Procrustes analysis [8]. The methods for point cloud registration do not, however, solve the merging problem, but leaves a map with double representations of matching points.

When it comes to fusion of individual maps, there are different ways to do this, but many of the methods are developed for concurrent mapping. Several of them have been created to perform collaborative visual SLAM. There are examples of collaborative visual SLAM that work for several units at once and are fast enough to run in real-time [9]. Many of these examples are focused on implementations in drones flying simultaneously. In these cases, parts of the pipeline are run on the platform, while parts are computed in the cloud. The map fusion is then based on a few keyframes, to decrease the need for storage space [10]. There are also several examples where the bundle adjustment is only performed locally to decrease the computational effort [11]. In the collaborative SLAM method presented by [12] a dynamic environment is possible and several cameras can be used. However, the cameras are initialised by viewing the same scene. This simplifies the global coordinate system, but is not always applicable, since it is often the case that there are no common points for all bundle sessions. There are also studies where several cameras have been used at the same time, but when they are fixed on a stereo head [13]. All these methods are developed for simultaneous mapping using several cameras. When maps from different occasions are merged the conditions change, which gives other limitations and possibilities.

Another important problem within SfM is the ability to perform *loop closure*. This problem appears when a reconstruction is made iteratively on a long image sequence and some feature points reappear after some time. Due to the inherent drift and error accumulation the reappearing points will not be reconstructed at the same position as they where reconstructed initially. For the loop closure problem it is assumed that it is possible to identify which points in the images that belong to the same 3D point. When re-appearing points are detected, it is possible to utilise this information and increase the quality of the reconstruction and at the same time position these at the same 3D location. Some techniques for loop closure can be found in [14–16].

There are also examples where SLAM is solved using a Bayesian approach [17], which is faster but not as accurate as bundle adjustment [4]. The methods that are discussed so far in this paper are not only applicable to images, but work similarly for other sensor data as well, e.g. wifi [18] and audio [19, 20]. Gaining information and ideas from these fields can thus be useful for SfM as well.

In [21], a method that is a compromise between a full optimisation bundle and the Kalman filter was presented. The method was primarily evaluated on audio data together with a small experiment for image data. In this paper we develop that idea further to work automatically for SfM data from RGB images. The idea behind the method presented in [21] is that maps can be merged efficiently using only a small memory footprint from the map and the residuals. Then the merging problem can be solved linearly. For this to work on images from different datasets there is a need for a coordinate system estimator.

In this paper, we present a method for efficient and simultaneous estimation of the parameters — i.e. camera matrices and 3D points — as well as the coordinate system. The system is also adopted for partially non-overlapping data. The pre-process of the data starts with the detection of feature point descriptors, e.g. using SIFT [22] or ORB [23], whereupon the individual maps are estimated using a SfM pipeline [24]. The data can be collected using a hand-held camera or an autonomous drone, like the one in Figure 1. The individual maps are then fused at the same time as transformations into a global coordinate system are estimated using only the map points and a compressed representation of the Jacobian. This means that once the individual maps are computed, there is no need for saving the actual images — not even a few keyframes.

The main contribution of this paper is the generic and efficient method for merging sub-maps obtained from several image sequences. The proposed method is independent of the chosen order of the sequences and the choice of coordinate system. It is furthermore very efficient compared to making a full bundle adjustment of all image sequences together, by utilising a compact and efficient representation of each bundle, consisting of a considerably reduced number of free variables. The merging method can also be used to detect changes in a scene and to solve the loop closure problem. This is validated on both simulated and real data.

2 SfM Systems and Bundle Adjustment

The pre-processing steps, such as creating the individual map representations, are not the focus of this paper. Nonetheless, we will briefly go through the theory. Many of the notations that will be used later in the paper are introduced in this section. The purpose of the individual optimisation bundles is to find the m camera matrices P_i and the n 3D points U_j that induce the image points u_{ij}. Each image point gives rise to two residual terms r_{ij}, one for each image coordinate, when it is compared to the projection of U_j in camera i,

$$r_{ij} = \begin{bmatrix} \frac{P_i^1 U_j}{P_i^3 U_j} - u_{ij}^1 \\ \frac{P_i^2 U_j}{P_i^3 U_j} - u_{ij}^2 \end{bmatrix}. \tag{1}$$

Above, P_i^k denotes row k of camera matrix P_i and u_{ij}^k denotes element k of u_{ij}. The total residual vector $\mathbf{r}$ is composed by stacking all individual residual vectors r_{ij}. Furthermore, we collect the unknown parameters in a structure $\mathbf{z}$, s.t.

$$\mathbf{z} = (P_1,\, P_2,\, \ldots\, P_m,\, U_1,\, U_2,\, \ldots\, U_n). \tag{2}$$

In the optimisation we use local parametrisations, $\Delta\mathbf{z} \in \mathcal{R}^{6m+3n}$, around each point $\mathbf{z}_0$ in the parameter space,

$$(\mathbf{z}_0, \Delta\mathbf{z}) \longrightarrow \mathbf{z}. \tag{3}$$

While $\mathbf{z}$ contains twelve parameters for each camera, the local parametrisations only use six parameters per camera, in order to assure that the camera is composed by a rotation matrix and a translation matrix. To find out how a change $\Delta\mathbf{z}$ affects the residual $\mathbf{r}$, we compute the derivatives of the components in $\mathbf{r}$ with respect to the elements in $\Delta\mathbf{z}$. Even if we only refer to the six parameters per camera together with the 3D points we will for simplicity further on denote this Jacobian $J = \partial \mathbf{r} / \partial \mathbf{z}$. The maximum likelihood estimate $\mathbf{z}^*$ of $\mathbf{z}$ is found by minimising the sum of squared residuals,

$$\mathbf{z}^* = \operatorname{argmin}_{\mathbf{z}} \mathbf{r}^T \mathbf{r}. \tag{4}$$

Using Gauss-Newton, each step of the iterative bundle adjustment corresponds to the parameter update

$$\Delta \mathbf{z} = -(J^T J)^{-1} J^T \mathbf{r}. \tag{5}$$

Performing the optimisation on N separate data collections results in N different parameter representations $\mathbf{z}^{(k)}$, where superscript index (k) denotes the representation number. Some of the 3D points are visible in several representations while some are visible in only one. Note that the ordering might differ, such that $U_j^{(k)}$ does not represent the same point as $U_j^{(l)}$. Once matches between the different data collections have been established, e.g. using ORB or SIFT features, the individual map representations can be merged into one global map. One way to do this would be to perform a bundle adjustment with all data from all data collections, but this could be prohibitively expensive in terms of memory and computations. Another way could be to do co-registration of the point clouds, e.g. using Procrustes. The naive way to merge the maps would be to then take the average position of matching points. One drawback of this merging method is that the resulting global map is depending on the merging order.

2.1 A Compact and Efficient Model for a Bundle Session

The proposed method exploits the fact that the optimal residuals from the separate bundles can be linearised to avoid the large bundles. Our bundle representation is built on theory from [21] and for completeness, we will summarise some of that theory in this section.

A key idea is to divide the unknown parameters in $\mathbf{z}$ into two parts $\mathbf{q}$ and $\mathbf{s}$, where $\mathbf{q}$ contains the parameters that potentially could match to those of other SfM sessions. The parameters in $\mathbf{s}$ can be thought of as auxiliary parameters. There is an interesting trade-off here. Making $\mathbf{q}$ larger allows for a higher number of potential matches with other SfM sessions, but requires a large memory footprint and vice versa. In this paper we use the approach that some (or all) of the 3D points go into $\mathbf{q}$, whereas the rest of the points and camera matrices go into $\mathbf{s}$.

Approximating the Residual

The parameters in $\mathbf{z}$ are ordered such that $\Delta\mathbf{z} = \begin{bmatrix}\Delta\mathbf{q} & \Delta\mathbf{s}\end{bmatrix}^T$. The Jacobian J is divided correspondingly, with the part that corresponds to the parameters in $\mathbf{q}$ denoted J_a and one that corresponds to the parameters in $\mathbf{s}$ denoted J_b. The auxiliary parameters in $\mathbf{s}$ will depend on the points in $\mathbf{q}$ as follows

$$\frac{\partial\mathbf{s}}{\partial\mathbf{q}} = -(J_b^T J_b)^{-1}(J_a^T J_b)^T. \tag{6}$$

That derivative can furthermore be used to express how the residuals change if the points in $\mathbf{q}$ are moved. We have that

$$\Delta\mathbf{r} = \underbrace{\left(J_a + J_b \cdot \frac{\partial\mathbf{s}}{\partial\mathbf{q}}\right)}_{J_q}\Delta\mathbf{q}. \tag{7}$$

Furthermore, viewing the residual as a function of an update $\Delta\mathbf{q}$ and linearising it around an optimal point o gives the following approximation of the squared residual

$$\mathbf{r}^T\mathbf{r} \approx a^2 + \Delta\mathbf{q}^T R^T R \Delta\mathbf{q}, \tag{8}$$

where $a^2 = \mathbf{r}|_o^T \mathbf{r}_o$ and R is a triangular matrix originating from QR-decomposition of $J_q|_o$.

Another way to view this is to form a modified residual vector $\hat{\mathbf{r}}$ according to

$$\hat{\mathbf{r}} = \begin{bmatrix} a \\ R\Delta\mathbf{q} \end{bmatrix} = \begin{bmatrix} a \\ R(\mathbf{q} - \mathbf{q}|_o) \end{bmatrix}, \tag{9}$$

whose sum of squares is an approximation of the original sum of squares, i.e.

$$\mathbf{r}^T\mathbf{r} \approx \hat{\mathbf{r}}^T\hat{\mathbf{r}}. \tag{10}$$

The linearisation decreases the memory footprint substantially compared to the original problem.

To summarise the theory from [21], the compressed representation of data consists of $(\mathbf{q}|_o, a, R)$, where $\mathbf{q}|_o$ is a subset of the 3D points. Note that while J_q is a rectangular matrix, R will be quadratic and thus much smaller than J_q. Despite this, it was shown in [21] that it is possible to obtain a good approximation of the residual according to Equation (8). Furthermore, once an update has been made, the rest of the points and the camera matrices can be updated using $\partial\mathbf{s}/\partial\mathbf{q}$.

2.2 Gauge Freedom

SfM estimates can only be determined up to an unknown choice of coordinate system, which is called gauge freedom. This involves translation, rotation and change of scale, which in total has seven degrees of freedom. A consequence of this is that the Jacobian J_q and also the matrix R has a seven-dimensional nullspace. The process of changing coordinate system is however non-linear, and therefore the approximation we presented in the previous section is only valid for points close to the optimal point $\mathbf{q}|_o$. In [21] the different maps were pre-aligned and the gauge freedom was therefore less relevant.

3 Merging Several SfM Sessions

As we mentioned before, one way to add several individual SfM sessions would be to do a new bundle, over all data. However, this could be computationally expensive and require storing a large amount of data. The faster approach presented in [21] solved the problem linearly. Though, this did require that the individual map representations were aligned and that a number of points were visible in all maps. In this section, we generalise this further to work for any representations and handle the coordinate ambiguity. Thus, no pre-alignment is needed and that makes the approach much more flexible.

Assume that for each map k we have the compressed information as $(\mathbf{q}^{(k)}, a^{(k)}, R^{(k)})$. Denote the global map $\mathbf{q}$ and let that contain some or all of the 3D points that are contained in at least one of the individual maps $\mathbf{q}^{(k)}$.

If $\mathbf{q}$ is assumed to contain $\bar{n}$ 3D points, all potential global map representations lie on a $3\bar{n}$-dimensional manifold. Each representation can then be projected to lower dimensional spaces in which the local map representations lie. In practice, the projection $p_k(\mathbf{q})$ simply means that we leave some points out, while we keep the rest, i.e. $p_k(\mathbf{q})$ has the same number of points (and the same point order) as $\mathbf{q}^{(k)}$. However, they might be in a different coordinate system. Therefore, we apply a similarity transform T_k to obtain a local map representation. Hence, we would like $T_k p_k(\mathbf{q})$ to be close to $\mathbf{q}^{(k)}$.

Thence, the unknowns are the global map $\mathbf{q}$ and the N different transformations T_k. By collecting individual residuals, similar to the ones in (9), the approximate modified residuals

$$\hat{\mathbf{r}} = \begin{bmatrix} a^{(1)} \\ R^{(1)}(T_1 p_1(\mathbf{q}) - \mathbf{q}^{(1)}) \\ \vdots \\ a^{(N)} \\ R^{(N)}(T_N p_N(\mathbf{q}) - \mathbf{q}^{(N)}) \end{bmatrix} \tag{11}$$

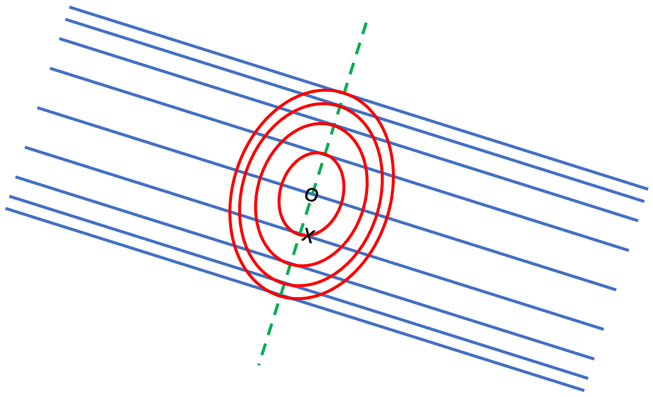


Figure 2: The blue solid lines show the level curves of the linearised error function and the o shows the point we have linearised around. By adding a penalty in the perpendicular direction, along the dashed green line, the resulting error function is the one shown as red ellipses.

are such that $\hat{\mathbf{r}}^T\hat{\mathbf{r}}$ is approximately equal to the sum of the squared residuals for all N sessions.

Our approach to solve this problem is to bundle over $\mathbf{q}$ and all T_k, trying to minimise (10) with $\hat{\mathbf{r}}$ according to (11). This bundle will be significantly smaller and faster than bundling over all the original images with the re-projection errors as loss.

One difficulty with this approach is that the approximation (8) only holds when each map $T_k p_k(\mathbf{q})$ is close to its working point $\mathbf{q}^{(k)}$. An interesting thing to note here is that the last seven rows of $R^{(k)}$ — the triangular matrix which comes from QR-decomposition of the Jacobian w.r.t. $\mathbf{q}$ of bundle k — will be zero, due to the gauge freedom. An illustration of this is given in Figure 2, where we visualise this in a lower dimension. The error function locally looks like a parabolic cylinder (illustrated with blue level curves in the figure). To force the error function to be quadratic — a paraboloid — we change the last seven rows of $R^{(k)}$ such that they are orthogonal to the rest of the rows (see the green dashed line). The change of the last rows of $R^{(k)}$ results in an error function that has the level curves shown by the red ellipses. By optimising over the transformation $T^{(k)}$, we will end up at a point where the orbit is tangent to a level curve (a red curve in Figure 2). One such point could be the one marked by x in the image. Since this x is close to our working point o, the linearisation is still valid and the addition of the last rows of $R^{(k)}$ will have little undesired effect.

3.1 The Bundle Initialisation

The initialisation for the merge bundle can be done in several ways. We have decided to initialise $\mathbf{q}$ from $\mathbf{q}^{(1)}$, and T_1 to be the identity matrix, while we initialise the rest of the T_k:s using Procrustes analysis between matching points in $\mathbf{q}^{(k)}$ and $\mathbf{q}^{(1)}$. Points in $\mathbf{q}$ that are not in $\mathbf{q}^{(1)}$ are initialised from the other maps and T_k. Initial estimates between two maps can be obtained using three point correspondences, but there are also solvers for mini-loop closure involving fewer than three points between three, four and five maps, cf. [25].

3.2 The Bundle for Merging the Maps

Similar to the individual bundle approach, we collect the unknown variables in a structure $\mathbf{w}$, s.t.

$$\mathbf{w} = (\mathbf{q},\ T_1,\ T_2,\ \ldots\ ,T_N), \tag{12}$$

and use local parametrisations, $\Delta\mathbf{w} \in \mathcal{R}^M$, around each point $\mathbf{w}_0$ in the parameter space,

$$(\mathbf{w}_0, \Delta\mathbf{w}) \longrightarrow \mathbf{w}. \tag{13}$$

The dimension M of $\Delta\mathbf{w}$ depends on how many of the points in $\mathbf{q}$ that are common between the individual maps. In the local optimisation we use a Levenberg-Marquart approach. In each step we calculate the Jacobian J_w that describes how changes in the parameters $\Delta\mathbf{w}$ affect the residual $\hat{\mathbf{r}}$.

3.3 Compressing the Result from the Merge

Once the merge is done, the residuals can be compressed for future merges. The Jacobian J_w is again divided in one part $\bar{J}_a$ that corresponds to the parameters in $\mathbf{q}$ and one $\bar{J}_b$ that corresponds to the rest of the parameters $\mathbf{s}$ (now corresponding to changes in $T_1, \ldots, T_N$ and some of the 3D points). From this we again calculate how $\mathbf{s}$ depends on $\mathbf{q}$, similar to what we did in Equation (6). We then calculate $\bar{R}$ from a QR-factorisation of $\bar{J}_q = \bar{J}_a + \bar{J}_b \cdot \partial\mathbf{s}/\partial\mathbf{q}$. In this way, a compact representation $(\mathbf{q}, \bar{a}, \bar{R})$ of the result can be calculated, again similar to what we did for the individual bundle sessions. The value $\bar{a}^2$ is the squared residual in the optimal point.

4 Hypothesis Testing of the Merge

Even if we assume that the matches between the maps are given, some of them might be wrong. Also, there could be other errors in the merge, e.g. if any object in the scene has moved. For this reason, some hypothesis test is needed. We use the same approach as in [21] and compare the increased error to a Γ distribution, and extend that theory here. Again, let N denote the number of map representations that are merged.

If we assume that the measurement errors in the images are zero mean Gaussian with a standard deviation σ, the expected value of the squared residuals $(a^{(k)})^2$ in the individually optimal points from (8) are

$$\mathbf{E}[(a^{(k)})^2] = \mathbf{E}[(\mathbf{r}^{(k)})^T(\mathbf{r}^{(k)})] = \sigma^2(\eta_{res}^{(k)} - d_{dof}^{(k)}). \tag{14}$$

We denote the number of residuals in bundle k by $\eta_{res}^{(k)}$ and the effective degrees of freedom $d_{dof}^{(k)}$. In a bundle with m cameras and n 3D points, these will be $\eta_{res} = 2mn$ and $d_{dof} = 6m + 3n - 7$, where the 7 represents the gauge freedom. Furthermore, if we assume that the merge was successful, the expected value for the merged map will be

$$\mathbf{E}[\bar{a}^2] = \mathbf{E}[\mathbf{r}^T\mathbf{r}] = \sigma^2(\eta_{res} - d_{dof}), \tag{15}$$

where d_{dof} is the effective degrees of freedom in the merge and η_{res} is the total number of residuals. We have that $\eta_{res} = \sum_k \eta_{res}^{(k)}$, while the value of d_{dof} will depend on the overlap between the individual map representations.

Now, for the difference between $\bar{a}^2$ and all $(a^{(k)})^2$, we have

$$\mathbf{E}\Big[\underbrace{\bar{a}^2 - \sum_k (a^{(k)})^2}_{\tilde{a}}\Big] = \sigma^2\Big(\eta_{res} - d_{dof} - \sum_k \big(\eta_{res}^{(k)} - d_{dof}^{(k)}\big)\Big). \tag{16}$$

Letting $\tilde{a} = \bar{a}^2 - \sum_k (a^{(k)})^2$ and denoting the number of 3D points that are common in i individual maps κ_i, this gives

$$\begin{aligned}\mathbf{E}[\tilde{a}] &= \sigma^2\Big(\sum_k d_{dof}^{(k)} - d_{dof}\Big)\\ &= \sigma^2\left(\Big(\sum_{i=1}^{N} 3\kappa_i(i-1)\Big) - 7\cdot(N-1)\right),\end{aligned} \tag{17}$$

where the factor $3\kappa(i-1)$ represents that we have locked another $3\kappa_i$ point coordinates $i-1$ times. We subtract by $7\cdot(N-1)$ since all individual maps are now merged to the same coordinate system.

Furthermore, since the noise is Gaussian, the value in (17) will be a sum of $\sum_k d_{dof}^{(k)} - d_{dof}$ Gaussian distributed variables. Altogether, this means that for a successful merge, $\tilde{a}$ should come from a Γ distribution with the following density [26]

$$f_{\alpha,\nu}(x) = \frac{1}{\Gamma(v)}\alpha^\nu x^{\nu-1}e^{-\alpha x}. \tag{18}$$

Here, Γ is the gamma function and

$$\alpha = \frac{1}{2\sigma^2}, \quad \nu = \frac{\left(\sum_{i=1}^{N} 3\kappa_i(i-1)\right) - 7 \cdot (N-1)}{2}. \tag{19}$$

Hence, if the merge for some reason was not successful, this could be discovered by comparing the value of $\tilde{a}$ to the expected Γ distribution. Using this, we can detect whether changes has occurred in the scene between the different mapping occasions.

The standard deviation σ of the noise is often unknown. Nevertheless, it can be estimated as the mean of the standard deviations for the individual map representations, which in turn would be estimated according to [27, p. 47].

5 Using Merging for Increased Robustness

Once we know which Γ distribution the increased error $\tilde{a}$ should come from, this can be used as a hypothesis test. This could furthermore be used to increase robustness in a large SfM session. If we do a bundle over a scene and the residuals are not sufficiently small, one might suspect that there is corrupt data or outliers involved. The SfM session could then be divided to a number of parts, where SfM first is performed on each of them, resulting in a number of sub-maps. Given that the different sub-maps are divided such that they have overlap, they can be merged using our method. By merging one part at a time and checking the distribution of $\tilde{a}$, it can be found where in the dataset there is corrupt data. That part can thereafter be divided into smaller parts, and the process can be repeated. We can by that avoid to add erroneous information to the global map, while we successfully can add the other parts which are correct.

6 Experimental Validation

To verify the proposed method we have run a number of experiments, both on simulated and real image data. The experiments are described in an order of increased complexity.

6.1 Verification on Simulated Data

First, we verified the method and the hypothesis test on simulated data. We simulated 100 3D points U_j in a box of size $10 \times 6 \times 2$ and ten cameras P_i pointing towards the box. The cameras were re-simulated three times to mimic three mappings. All 3D points were visible in all cameras and we added Gaussian noise with zero mean and standard deviation

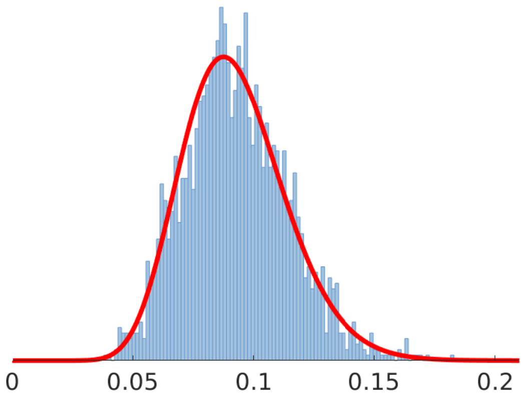


Figure 3: The image shows the histogram of $\tilde{a}$ achieved from running the same experiment several times with different noise realisations. The histogram is expected to follow the Γ distribution shown by the red curve.

$\sigma = 0.05$ in the image projections. On each mapping we separately performed bundle adjustment using only the image projections u_{ij}, resulting in three different representations of the same scene, each given in a different coordinate system. We found matches between all three map representations and using ten of these matches in $\mathbf{q}$, we merged the maps into one global map.

We repeated the experiment above 2 000 times with the same cameras and 3D points, but with different noise realisations. For each run, we saved $\tilde{a}$ and finally we plotted a histogram over the result. Figure 3 shows the histogram together with the expected Γ distribution from Equation (18). The figure clearly shows that the error follows the distribution even though we have linearised the residual according to Equation (11). This is a verification that the method works even when we are optimising over the transformations to the global coordinate system as well as the global map.

Early Stopping of Pre-Processing

In the previous experiment we let the individual map representations reach an optimal state before merging them. However, this is not always the case in reality, due to poorly chosen bundle thresholds or to shortage of time. In the second experiment we investigated how the system performance degrades with less optimisation in the pre-processing steps. The setup was similar; all 3D points were visible in all map representations (but only in 80 % of the cameras). We used noise with $\sigma = 0.005$. We stopped the bundle for the individual maps based on the Euclidean norm of the gradient $2\hat{\mathbf{r}}J$, which is obtained by differentiating (10), with the residual given by (9) and the Jacobian J defined from that. The termination was set

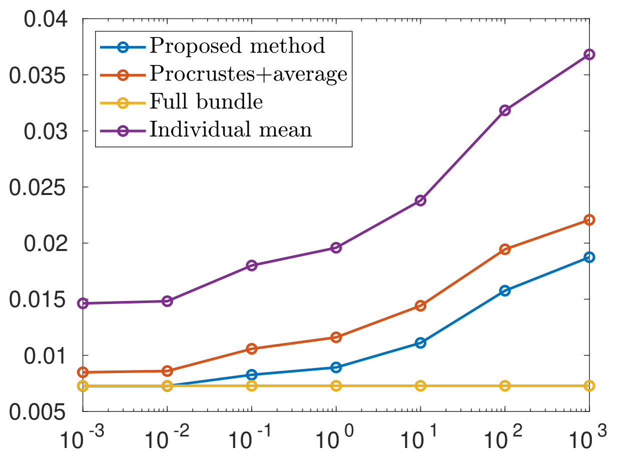


Figure 4: How the RMSE for the final map achieved using different merging methods change when the individual map bundles are terminated at different levels. The x-axis shows the Euclidean norm of the gradient and the y-axis the RMSE. The error for the individual maps is included for comparison.

at different levels, after which we used our proposed method for merging. For comparison, we performed a large bundle on all data, and we also did Procrustes registration (using an arbitrary order) followed by averaging over matching points. The mean RMSE over 1 000 runs was computed and is plotted in Figure 4. For comparison, the mean RMSE for the three individual map representations is shown as well. To compute the RMSE we first did Procrustes registration of the respective map to the true map.

Our proposed method performs better than Procrustes at all stages, and furthermore the graph is less steep than that of the individual errors, which means that some of the performance that is lost from the early stopping is recovered using our method. Finally, one can see that for small gradient norms, our method performs as well as the large bundle, which is much more computationally expensive.

Solving Loop Closure by Map Splitting

Furthermore, we wanted to show that our method can solve the problem of loop closure, not within one individual bundle, but in the merging of several slightly overlapping sessions. We also decreased the number of parameters in $\mathbf{q}$ to be a small part of all the 3D points. First off, we simulated a SfM session of a room of size $5 \times 6 \times 2$ m and divided it into four parts, such that each sub-map captured one of the walls, with a few corner points common between the different sub-maps, and no points common in more than two maps. Each sub-map consisted of 200 3D points and only 6 % of these coincided with points from any of the other maps. The noise level was $\sigma = 0.005$.

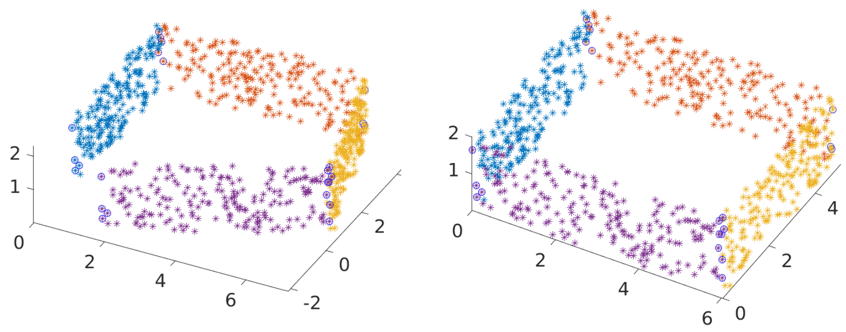


Figure 5: The left plot shows the result from merging the different sub-maps without matches in the beginning and the end of the map. To the right we have added matches between these two and the loop closure problem is solved.

To simulate a loop closing problem we used two merging methods — one with matches between sub-maps 1-2, 2-3 and 3-4 and one where there were matches between sub-maps 1-4 as well. The first case represents what happens when you do SfM starting at one point of the room and do a loop without using any loop closing technique, while the latter uses our proposed method. We could see that our method improves the performance concerning loop closure. In most cases there was a drift in the map for the first method, but if we added matches between sub-maps 1-4 as well, this drift disappeared. Figure 5 shows how the method fails to connect the ends of the blue and the purple sub-maps in the first case, but succeeds in the second.

Running the same experiment 1 000 times shows that adding matches between sub-maps 1-4 gives a reduced Euclidean distance from the ground truth in 85 % of the cases, and the distance is reduced by 50 % or more in 80 % of the cases. In terms of RMSE, this error was less than 0.1 in 99.8 % of the cases for the full bundle. This can be considered as gold standard. The corresponding value for our method with all matches was 80 %; for our method without 1-4 matches 25 %; and for Procrustes and averaging 7.7 %. The mean RMSE within those 80 % for our method was 0.038. If the merge is unsuccessful, the RMSE value is not very suggestive, since the registration made for comparison might be wrong too.

Furthermore, Table 1 illustrates how much the memory footprint is decreased when we use our compressed error representation. The linearised residual in (8) reduces the parameters in the Jacobians from approximately $3\,000 \times 660$ to 30×30 compared to the full residual. This becomes even more evident when we look at the size of the bundle for the map merging, compare Equations (10) and (11). All this show that our proposed method performs best except for the full bundle and that it therefore is a very good compromise between performance and efficiency.

Table 1: Four sub-maps were merged. The table shows the size of the Jacobian for using the full residual (1) and the linearised residual (8) for each sub-map. The last line shows how much smaller the merging problem becomes using our method.

Bundle session	# points	Size of full Jacobian	Size of compressed Jacobian
1	200	$3\,082 \times 660$	27×27
2	200	$2\,792 \times 660$	24×24
3	200	$3\,140 \times 660$	33×33
4	200	$3\,190 \times 660$	36×36
merge	784	$12\,204 \times 2\,601$	120×88

6.2 Verification on Real Data

Small Bookshelf Experiment

In this experiment we made five separate data collections of a bookshelf. Between collection 2 and 3 we moved an R2D2 figure a few centimeters, see Figure 6. The individual maps were then merged pairwise in sequence — i.e. 1-2, 2-3, 3-4 and 4-5 — and compared with our previous work [21] where the transforms between maps were computed prior to the merging. As we see in Figure 7 the residuals are smaller when jointly estimating merge and transform. The squared residuals $\tilde{a}$ are then compared with the 99:th percentile of the Γ distribution from Section 4. We see that change between collection 2 and 3 is correctly detected for both versions, while the previous work with fixed transform is giving a false positive between dataset 3 and 4. Even if the differences are small, this experiment shows that our proposed method performs better than the previous one, despite the problem being harder.

Experiment in an Office Environment

In the following experiment we made four separate data collections using a drone. Sample images from these datasets, as well as 3D reconstructions, are shown in the two top rows of Figure 8. Each recording consisted of approximately a minute worth of video footage. The recordings were made with a small drone equipped with a monochrome global shutter camera (OV9281) with resolution 480×640 and an inertial measurement unit (MPU-9250). The 3D reconstructions were generated by a SLAM system built on ORB features [23] and IMU data [28], where the matches are filtered using the technique from [29] and the solution is optimised using [30]. For each of the reconstructions, the object points were saved along with extracted feature locations and descriptors. The feature locations were undistorted prior to saving, to remove fish-eye effects.

The statistics for the four experiments are shown in Table 2. The saved descriptors were used to generate hypothesis matches between the different reconstructions. These tentative

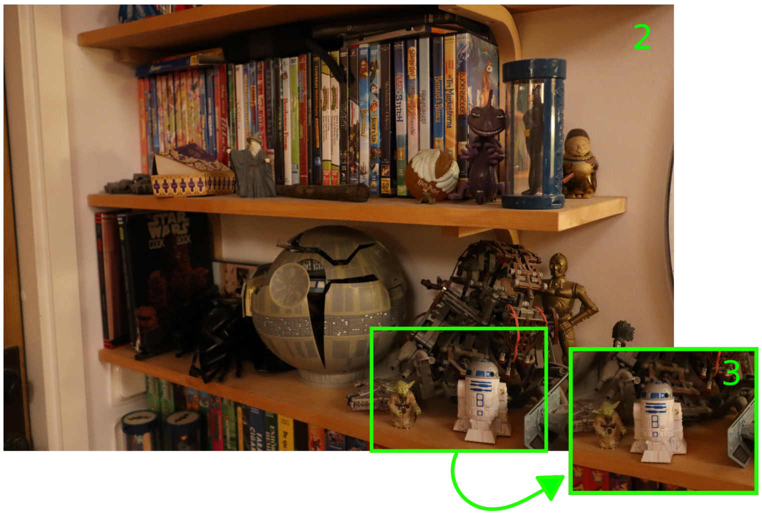


Figure 6: This figure shows how the R2D2 model moved between collection 2 and 3 in the bookshelf experiment.

matches were then tested in a hypothesis and testing framework using the hypothesis test proposed in Section 4 of this paper. This process produced 24 points that were matched across the four experiments.

In the bottom row of Figure 8 we show parts of the merged map after Procrustes to the left and merging using our method to the right. Notice that the top and left wall in the upper left corner had double representations after Procrustes. After merging the two copies of the walls they are positioned on top of each other. After the merge with the proposed method it was possible to identify an additional 346 points that could be merged.

To validate the performance we selected a few points in one of the maps and calculated a number of interpoint distances before and after merging. We also measured these distances in reality with a measuring tape. The results are presented in Table 3. The results show that our method reduces the error in all the measured distances.

7 Conclusion

In this paper we have presented a new method for merging of 3D maps. The method relies on a low memory footprint representation of the individual residuals that makes it efficient even for a large amount of image data. By bundling over an approximate error, the size of the Jacobian is reduced with several orders of magnitude compared to doing bundle adjustment over all data at once. Furthermore, the method is robust and flexible in the sense that the individual sub-maps do not have to be in the same coordinate system.

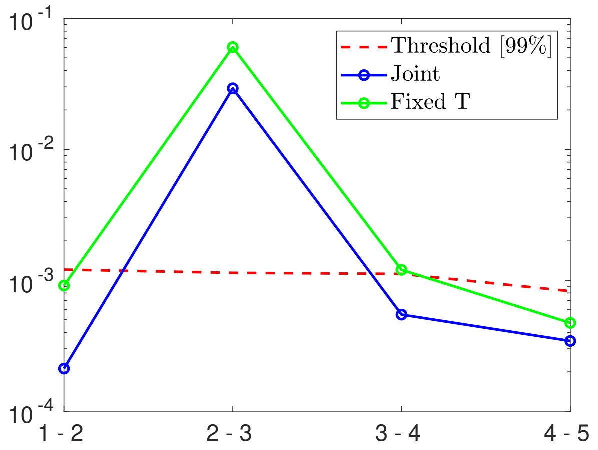


Figure 7: The sum of squared residuals for the merges of different dataset pairs. We see that jointly estimating the transform and 3D points during the merge yields smaller residuals than when estimating the transform before merging as in previous work [21]. Change between dataset 2 and 3 is correctly detected.

Our merging method can be used to add two or several maps at once and also for updating a global map using local map estimates. This can furthermore be used to perform loop closing, which is verified using both simulated and real data. Using a hypothesis test based on a statistical analysis of the error we can analyse whether the merge was successful and discover if changes has occurred in the scene between the mappings. In the future we would like to use this to develop a system that can divide a large map into several sub-maps in order to only add the parts of the map that preserves robustness. Another interesting extension would be to generalise the method to rotation averaging.

Table 2: Four datasets were collected by drone recordings. The number of 3D points and the size of the Jacobians for each dataset are shown. The proposed method makes it possible to compress the data to a 72×72 matrix for each dataset.

Bundle session	# points	Size of full Jacobian	Size of compressed Jacobian
1	999	$18\,918 \times 3\,621$	72×72
2	603	$11\,972 \times 2\,151$	72×72
3	549	$11\,114 \times 1\,989$	72×72
4	386	$7\,596 \times 1\,452$	72×72
merge	2465	$49\,600 \times 8\,997$	288×100

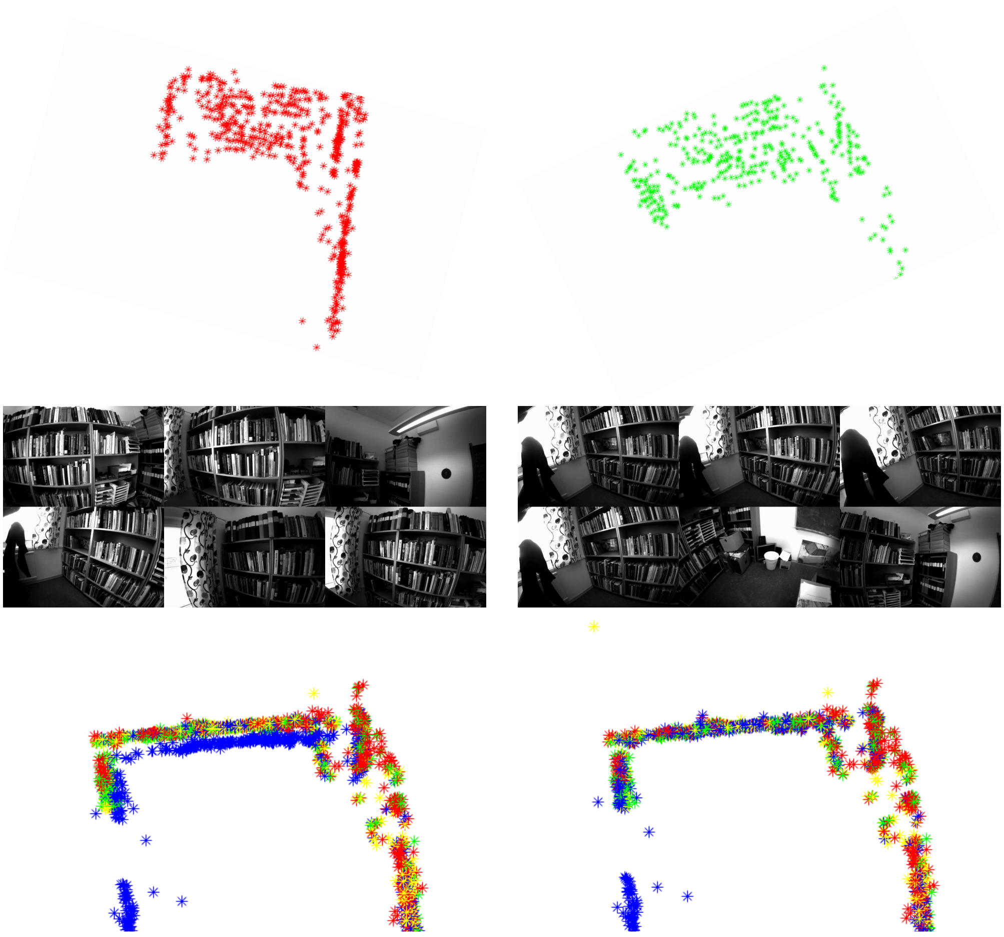


Figure 8: The two top rows shows the 3D reconstructions and a few images from two of the four drone recordings in the office experiment. The bottom row shows parts of the merged map using Procrustes to the left and our proposed method to the right. Note that the top and left walls are doubled after the Procrustes registration, while our method solves that problem.

Table 3: Interpoint distances between a few selected points in the office experiment before and after merging using Procrustes registration followed by averaging and our proposed method. The column to the right shows the ground truth distances.

Pt 1 ind	Pt 2 ind	Dist (mm) one map	Dist (mm) merge Pro.	Dist (mm) merge our	Dist (mm) gt
52	766	365	365	220	213
52	839	589	589	512	516
52	840	1358	1296	1264	1260
60	839	825	825	834	840
60	840	879	1023	860	857

References

[1] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping (slam): Part i the essential algorithms," *Robotics and Automation Magazine*, vol. 13, no. 2, 2006.

[2] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, 2007.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge university press, 2003.

[4] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment — a modern synthesis," in *Proc. International Workshop on Vision Algorithms*. Springer, 1999.

[5] C. Engels, H. Stewénius, and D. Nistér, "Bundle adjustment rules," *Photogrammetric Computer Vision*, vol. 2, 2006.

[6] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *Proc. IEEE and ACM International Symposium on Mixed and Augmented Reality*, 2007.

[7] Y. Chen and G. G. Medioni, "Object modeling by registration of multiple range images." *Image and Vision Computing*, vol. 10, no. 3, pp. 145–155, 1992.

[8] D. G. Kendall, "A survey of the statistical theory of shape," *Statistical Science*, pp. 87–99, 1989.

[9] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 6, 2007.

[10] P. Schmuck and M. Chli, "Multi-uav collaborative monocular slam," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[11] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyser, and P. Sayd, "Real time localization and 3d reconstruction," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.

[12] D. Zou and P. Tan, "Coslam: Collaborative visual slam in dynamic environments," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 2, 2013.

[13] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004.

[14] P. Newman and K. Ho, "Slam-loop closing with visually salient features," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[15] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardos, "An image-to-map loop closing method for monocular slam," in *Proc. International Conference on Intelligent Robots and Systems (IROS)*, 2008.

[16] N. Guilbert, M. Kahl, M. Oskarsson, K. Åström, A. Heyden, and M. Johansson, "Constraint enforcement in structure and motion applied to closing and open sequence," in *Proc. Asian Conference on Computer Vision (ACCV)*, 2004.

[17] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2003.

[18] K. Batstone, M. Oskarsson, and K. Åström, "Robust time-of-arrival self calibration and indoor localization using wi-fi round-trip time measurements," in *Proc. IEEE International Conference on Communications Workshops (ICC)*, 2016.

[19] Y. Kuang, S. Burgess, A. Torstensson, and K. Åström, "A complete characterization and solution to the microphone position self-calibration problem," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[20] S. Zhayida, F. Andersson, Y. Kuang, and K. Åström, "An automatic system for microphone self-localization using ambient sound," in *Proc. IEEE European Signal Processing Conference (EUSIPCO)*, 2014.

[21] G. Flood, D. Gillsjö, A. Heyden, and K. Åström, "Efficient merging of maps and detection of changes," in *Proc. Scandinavian Conference on Image Analysis (SCIA)*. Springer, 2019.

[22] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *Proc. IEEE International conference on computer vision (ICCV)*, 2011.

[24] C. Olsson and O. Enqvist, "Stable structure from motion for unordered image collections," in *Proc. Scandinavian Conference on Image Analysis (SCIA)*. Springer, 2011.

[25] P. Miraldo, S. Saha, and S. Ramalingam, "Minimal solvers for mini-loop closures in 3d multi-scan alignment," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[26] W. Feller, *An Introduction to Probability Theory and Its Applications*. John Wiley & Sons, 1968, vol. 2.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference, and prediction*, 2nd ed. Springer Science & Business Media, 2009.

[28] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation." Georgia Institute of Technology, 2015.

[29] G. Vogiatzis and C. Hernández, "Video-based, real-time multi-view stereo," *Image and Vision Computing*, vol. 29, no. 7, 2011.

[30] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, 2012.

Paper VII

Minimal Solvers for 3D Map Matching with Statistical Deformations

Gabrielle Flood, Erik Tegler, David Gillsjö, Anders Heyden and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: An important issue in simultaneous localisation and mapping is how to match and merge individual local maps into one global map. This is addressed within the field of robotics and is crucial for multi-robot SLAM. There are a number of different ways to solve this task depending on the representation of the map. To take advantage of matching and merging methods that allow for deformations of the local maps it is important to find feature matches that capture such deformations. In this paper we present minimal solvers for point cloud matching using statistical deformations. The solvers use either three or four point matches. These solve for either rigid or similarity transformation as well as shape deformation in the direction of the most important modes of variation. Given an initial set of tentative matches based on, for example, feature descriptors or machine learning we use these solvers in a RANSAC loop to remove outliers among the tentative matches. We evaluate the methods on both synthetic and real data and compare them to RANSAC methods based on Procrustes and demonstrate that the proposed methods improve on the current state-of-the-art.

Keywords: Structure from motion, Map Merging, Minimal Solvers, Statistic Deformation

1 Introduction

Simultaneous localisation and mapping (SLAM) and structure from motion (SfM) refer to the process of estimating map parameters as well as sensor motion, using only sensor data [12, 13]. In this paper we will be focusing on image data, but the methods presented are applicable to other sensor modalities, such as RGB-D cameras, lidar, microphones and radio receivers. If the reconstruction of a scene is made from a long image sequence and the camera returns to a position that previously has been visited there is a risk that the reconstruction of a point does not end up at the same position as it did when it was reconstructed earlier. This is due to the inherent drift and error accumulation in the SLAM system and is referred to as a loop closing problem. To solve this problem, these point must be detected as duplicates.

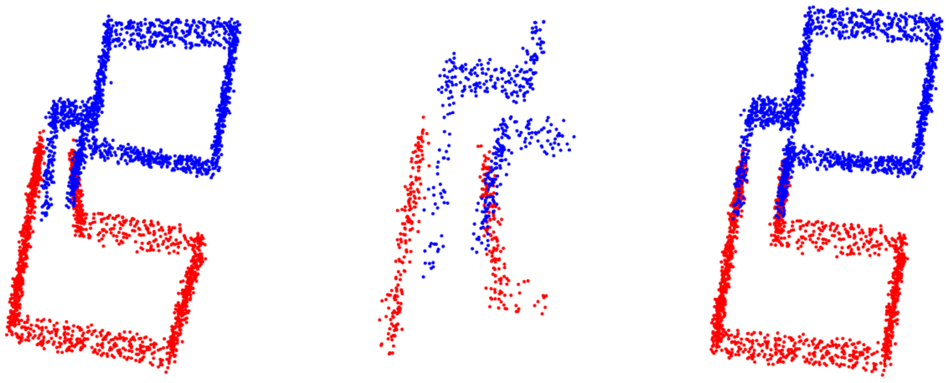


Figure 1: The figure shows a motivating example, where the red and blue maps to the left should be matched. Since the overlapping area contains drift, a rigid matching method will not be able to find all the matches, see the zoomed in image in the middle. Allowing for statistical deformations, such matches can be found and the errors can be corrected, as in the right plot.

A similar problem arises when several robots are working in an environment simultaneously. This is often referred to as colloborative SLAM [51] or multi-robot SLAM [50]. Each robot has its own SLAM system, building its own local map of the environment. To be able to locate each other and to cooperate in building one accurate, global map, these local maps need to be matched and merged. This will also occur when a single robot later returns to a previously reconstructed scene. Again, in order to do the merging, a reliable set of matches needs to be found. Yet another use-case is when a large dataset is split into smaller pieces, which are then merged. This allows for weeding out parts of the data that have produced a bad structure and motion estimate and thus producing a more robust result, see [52, 53].

Given a set of tentative matches, we present several minimal solvers that can be used to find a good inlier set. Many previous methods for point cloud matching only give rigid registrations. However, to capture loop closing issues and to match local maps that contain errors, this is not enough. Therefore, we allow for some statistical deformations in the direction of the major modes of variation – i.e. deformations of the 3D model that influence the reprojection error the least. Together with Random Sample Consensus (RANSAC) [14] we show that our solvers find a better inlier set compared to three and four point Procrustes [8, 21, 38, 39]. We then use this set in a map merging algorithm that allows for statistical deformations and show that local errors are corrected. The idea for the method is illustrated in Figure 1. The main contributions of this paper are[1]

- New minimal solvers for matching of 3D point clouds of known scale, using three or four point matches, which take statistical deformations in the direction of the most important modes of variation into account.

[1]Code: `https://github.com/gabrielleflood/statistical-mapmatching-minsolv/`

- New minimal solvers for matching of 3D point clouds of unknown scale, again using statistical deformations and either three or four point matches.
- A solver for matching of duplicate points within maps, using the modes of variation, to do, e.g. loop closing.
- A framework for how to use these new minimal solvers to match 3D point cloud maps.

2 Related Work

There are different representations of 3D reconstructions, e.g. surface based [19], voxel based and graph based methods, occupancy grids and point clouds [3, 47]. Here we will focus on point feature maps. There are several examples of feature detectors for images, e.g. SIFT [27], SURF [4], BRIEF [9] and ORB [35]. Knowing the features of the image points, these can be propagated to 3D (see Section 5). These features can then be used to find accurate matches between the maps.

The problem of finding the transformation that matches two points clouds optimally under the assumption of equal and independent Gaussian noise is a classical problem and the Procrustes algorithm [8] forms the backbone of many matching algorithms. This algorithm does neither handle outliers nor different error characteristics in the matching process [38]. The problem of finding an optimal transformation for a robust error norm is significantly more challenging. Nevertheless, there are polynomial time algorithms for solving this problem [31]. In [28] minimal solvers for line intersections with plane and point matches are presented. Other examples of solvers that work directly on point clouds are [29] which use fewer point correspondences but several maps and the 4PCS algorithm where coplanar sets of point points are used [1]. For all these methods it is common to use RANSAC loops to find robust matches and for many of them a set of tentative point matches are given [14]. However, few of the above algorithms handle the large scale uncertainties that one typically gets from SLAM or SfM estimates of maps. The 3D point matching problem has also been investigated using deep learning methods. One such example is the 3DRegNet, which both defines point matches as inliers or outliers and aligns the two scans [32]. Some other methods using learning are [34, 45, 46, 49].

For large scale 3D point cloud matching, the iterative closest point (ICP) algorithm and its many variants are commonly used [5, 10, 36]. However, these methods do not take advantage of any given point matches. In [26] the 3D point cloud is instead transformed into 2D bearing angle images, after which SURF features are used to find matching pixels, which also gives corresponding pairs in 3D. There are also map matching algorithms that use some a priori information to find matches between the maps, as in [33]. It is also common to initialise by letting all robots view the same scene for collaborative SLAM [51]. There are examples of SLAM systems that can merge parts of an active map when doing loop closure. Both [18, 30] are methods that perform loop closure based on bag-of-words for selected keyframes. Though, only rigid transformations are found. In [43, 44] non-rigid transformations are allowed for. Both are inspired by [40], where the map is divided into small clusters, which are aligned using affine transformations. However, this does not allow for different transformations for neighbouring points.

Matching of separate maps followed by map merging is in general commonly used for multi-robot systems [3, 47]. One way to do the matching between maps is by saving keyframes for each map and do an efficient search among these [41]. Map merging can also be used to divide mappings in smaller parts which are first reconstructed separately and then merged to decrease drift. Similarly, matching algorithms can be used to have more information in the form of a dense map in parts of a larger, sparse map [2].

Several review papers about map merging for multi-robot systems have been published during the last years and the authors conclude that more research on map merging in 3D is needed [3, 7] and that a key step in the merging is to extract stable point features [47]. Many of the matching and merging methods above only work with rigid transformations. If the reconstruction of a local map is erroneous there is no way to correct this later. The software package Maplab [37] contains tools for both building and merging maps – including matching them – captured at different occasions. For map merging they suggest rigid co-registration followed by iterative loop closing and bundle adjustment. There are some works that use statistical information, e.g. in [48], where principal component analysis is used to improve the initial matching for the ICP algorithm. In [24] the authors study the registration of two 2D maps with uncertainty and unknown transformation. It is shown that an approximation of the error function can be solved in closed form. In [15, 16], the authors develop methods that iteratively can solve for unknown transformations and exploit the uncertainty in the maps, but the problem of finding the correspondences is not considered.

3 Point Matching in Uncertain 3D Point Clouds

To address the difficulties with uncertainties in local 3D maps we propose minimal solvers that allow the points to deform in the direction of the most important modes of variation. We believe that such solvers are important components that can be used together with hypothesis and test paradigms and approximate merging algorithms [15, 16] or full bundle adjustement [54] in order to produce fast and robust map merging algorithms. Assuming that tentative point correspondences between maps and/or within a map are given and that we know the covariance of the point clouds – typically from bundle adjustment – we suggest to allow a deformation of the points in the direction of the modes of variation. Denote a point cloud (or map) $\mathcal{P}$ and the corresponding covariance $\mathbf{C}$. When doing eigenvalue decomposition of $\mathbf{C}$ there will always be seven eigenvalues that are infinite, due to gauge freedoms, since the rotation, translation and scale of the point cloud can be altered without affecting the residuals. Ignoring the eigenvectors corresponding to these seven eigenvalues, we denote the eigenvector corresponding to the $7+k$th largest eigenvalue $\mathbf{B}_k$. Henceforth, $\mathbf{B}_k$ will be referred to as the kth largest mode of variation.

We will first go through the different problems that we are interested in solving and then we will explain how the actual solvers are generated in Section 3.4.

3.1 One-to-One Matches within Point Clouds

When we assume that tentative point correspondences are given this also covers matches within the same map, i.e. loop closure or duplicate points. To achieve this, we transform the points along the K largest modes of variation

$$\mathbf{p}_i + \sum_{k=1}^{K} b_k \mathbf{B}_{ki} = \mathbf{p}_j + \sum_{k=1}^{K} b_k \mathbf{B}_{kj}, \tag{1}$$

where $\mathbf{B}_{ki}$ and $\mathbf{B}_{kj}$ denote the parts of $\mathbf{B}_k$ that correspond to the ith and jth map point, respectively, and $b_1, \ldots b_K$ are unknown factors. To match the points $\mathbf{p}_i$ and $\mathbf{p}_j$ we solve for these b:s and by adding exactly three modes we get a minimal system with one solution – a system of three linear equations with three unknowns. The idea is to minimise the negative log-likelihood of the new point cloud, while merging the points. This leads to the following optimisation problem

$$\begin{cases} \min\limits_{b} & \sum\limits_{k} \dfrac{b_k^2}{\lambda_k}, \\ \text{s.t} & \mathbf{p}_i + \sum_k b_k \mathbf{B}_{ki} = \mathbf{p}_j + \sum_k b_k \mathbf{B}_{kj}, \end{cases} \tag{2}$$

where λ_k is the eigenvalue corresponding to the kth mode of variation $\mathbf{B}_k$. The objective function in (2) is quadratic and the constraints are linear, hence the problem is easily solved.

3.2 Matches Between Point Clouds with Known Scale

To achieve a match between two different point clouds we present four minimal solvers, using either three or four point correspondences in maps where the scale is either known or unknown. The case with known scale is applicable, e.g. when the point clouds have been created using a SLAM system that incorporates IMU data or the point clouds have been scaled using some known distances within the maps.

Given that the scale in the two maps is known and the same, rigid registration would correspond to finding an unknown rotation and translation, i.e. six degrees of freedom. Although two point correspondences give six constraints, they can only be used to estimate five out of the six degrees of freedom in the similarity transform, since one constraint is undetermined.

Three Point Matching

The most common way to solve the registration problem is by using three point correspondences. If the point triplet $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \in \mathcal{P}$ should match $(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3) \in \mathcal{Q}$ that gives nine constraints. Hence, with the six degrees of freedom from the transformation we can solve for another three variables; thus we add three modes of variation. These modes can be added to either the first or the second triplet. If we call the modes of the first map $\mathbf{B}$ and those of the second $\bar{\mathbf{B}}$ this gives the modified points

$$\begin{aligned} \tilde{\mathbf{p}}_i &= \mathbf{p}_i + \sum_{j=1}^{k} b_j \mathbf{B}_{ji}, \quad i = 1,2,3, \\ \tilde{\mathbf{q}}_i &= \mathbf{q}_i + \sum_{j=1}^{3-k} \bar{b}_j \bar{\mathbf{B}}_{ji}, \quad i = 1,2,3, \end{aligned} \tag{3}$$

where k is an integer between 0 and 3, deciding how many modes of variation that are added to each triplet. We will mostly use $k = 1$, but all cases have been investigated. The rotation and translation between the maps can be eliminated to achieve three equations in the three unknowns b_1, $\bar{b}_1$ and $\bar{b}_2$ by first creating the vectors

$$\begin{aligned} \mathbf{v}_1 &= \tilde{\mathbf{p}}_2 - \tilde{\mathbf{p}}_1, \qquad & \mathbf{v}_2 &= \tilde{\mathbf{p}}_3 - \tilde{\mathbf{p}}_1, \\ \mathbf{w}_1 &= \tilde{\mathbf{q}}_2 - \tilde{\mathbf{q}}_1, \qquad & \mathbf{w}_2 &= \tilde{\mathbf{q}}_3 - \tilde{\mathbf{q}}_1, \end{aligned} \tag{4}$$

and then requiring that the triplet-shapes are the same

$$\begin{cases} \mathbf{v}_1^T \mathbf{v}_1 = \mathbf{w}_1^T \mathbf{w}_1, \\ \mathbf{v}_2^T \mathbf{v}_2 = \mathbf{w}_2^T \mathbf{w}_2, \\ \mathbf{v}_1^T \mathbf{v}_2 = \mathbf{w}_1^T \mathbf{w}_2. \end{cases} \tag{5}$$

Solving this system gives the three unknowns b_1, $\bar{b}_1$ and $\bar{b}_2$.

Four Point Matching

There are also examples of registration and matching methods that use four point correspondences. This utilises more information and in our case it allows for more modes to be used. Looking at the point quadruplets $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3, \mathbf{p}_4) \in \mathcal{P}$ and $(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3, \mathbf{q}_4) \in \mathcal{Q}$, this gives us in total twelve constraints, so we can solve for a total of six unknowns except for the rotation and translation. Similar to before, we modify the points,

$$
\begin{aligned}
\tilde{\mathbf{p}}_i &= \mathbf{p}_i + \sum_{j=1}^{k} b_j \mathbf{B}_{ji,}, \quad i = 1, \dots, 4, \\
\tilde{\mathbf{q}}_i &= \mathbf{q}_i + \sum_{j=1}^{6-k} \bar{b}_j \bar{\mathbf{B}}_{ji,}, \quad i = 1, \dots, 4,
\end{aligned}
\tag{6}
$$

where k is an integer between 0 and 6.

As in the three point case, we form the vectors

$$
\begin{aligned}
\mathbf{v}_i &= \tilde{\mathbf{p}}_{i+1} - \tilde{\mathbf{p}}_1, \quad i = 1, 2, 3, \\
\mathbf{w}_i &= \tilde{\mathbf{q}}_{i+1} - \tilde{\mathbf{q}}_1, \quad i = 1, 2, 3,
\end{aligned}
\tag{7}
$$

and eliminate the translation and rotation by creating the equations

$$
\mathbf{v}_i^T \mathbf{v}_j = \mathbf{w}_i^T \mathbf{w}_j, \quad 1 \leq i \leq 3,\ i \leq j \leq 3. \tag{8}
$$

3.3 Matches Between Point Clouds with Unknown Scale

Despite the possibilities to have a known scale of the point clouds, this is not always the case. Therefore, we have expressed three and four point solutions for unknown scale as well.

Three Point Matching

Again, we are looking at the point triplets $(\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3) \in \mathcal{P}$ and $(\mathbf{q}_1, \mathbf{q}_2, \mathbf{q}_3) \in \mathcal{Q}$ which give nine conditions. However, we can only solve for two additional parameters if the scale is unknown. Hence, we add two modes of variation to the triplets. This is done in the same way as in (3), but now k can be at maximum 2 and the second summation goes to $2 - k$. The vectors from (4) are created in the same way as before. However, when expressing the matching criteria, the scale has to be eliminated as well. This is done by taking the product of scalar products in the different triplets

$$\begin{cases} (\mathbf{v}_1^T\mathbf{v}_1)\cdot(\mathbf{w}_2^T\mathbf{w}_2) = (\mathbf{v}_2^T\mathbf{v}_2)\cdot(\mathbf{w}_1^T\mathbf{w}_1) \\ (\mathbf{v}_1^T\mathbf{v}_1)\cdot(\mathbf{w}_1^T\mathbf{w}_2) = (\mathbf{v}_1^T\mathbf{v}_2)\cdot(\mathbf{w}_1^T\mathbf{w}_1) \\ (\mathbf{v}_2^T\mathbf{v}_2)\cdot(\mathbf{w}_1^T\mathbf{w}_2) = (\mathbf{v}_1^T\mathbf{v}_2)\cdot(\mathbf{w}_2^T\mathbf{w}_2). \end{cases} \tag{9}$$

Four Point Matching

The equations for the four point solver with unknown scale will be expressed in a similar fashion. As in the three point case, we can add one less mode of variation compared to when the scale is known, i.e. a total of five modes. The modified points are given by (6), but with $k \leq 5$ and the upper bound of the second equation as $5 - k$. The vectors are formed according to (7). We further collect the scalar products within each quadruplet in a vector

$$\begin{aligned} \mathbf{o} &= [\mathbf{v}_1^T\mathbf{v}_1 \;\; \mathbf{v}_1^T\mathbf{v}_2 \;\; \mathbf{v}_1^T\mathbf{v}_3 \;\; \mathbf{v}_2^T\mathbf{v}_2 \;\; \mathbf{v}_2^T\mathbf{v}_3 \;\; \mathbf{v}_3^T\mathbf{v}_3], \\ \mathbf{u} &= [\mathbf{w}_1^T\mathbf{w}_1 \;\; \mathbf{w}_1^T\mathbf{w}_2 \;\; \mathbf{w}_1^T\mathbf{w}_3 \;\; \mathbf{w}_2^T\mathbf{w}_2 \;\; \mathbf{w}_2^T\mathbf{w}_3 \;\; \mathbf{w}_3^T\mathbf{w}_3]. \end{aligned} \tag{10}$$

Here, $\mathbf{o}$ can be seen as a vector collecting the unique elements of the Gram matrix of the vectors $\mathbf{v}_i$. If we let o_i denote the ith entry of $\mathbf{o}$ and correspondingly for $\mathbf{u}$, we obtain 15 equations

$$o_i \cdot u_j = o_j \cdot u_i, \quad 1 \leq i \leq 6,\; i < j \leq 6. \tag{11}$$

3.4 The Minimal Solvers

In this paper we study solvers for five problems, described by (2), (5), (8), (9), (11). Of these the problem described by (2) is easy to solve. State-of-the-art methods to handle the resulting systems of polynomial equations use theory from algebraic geometry, e.g. action matrix methods [11], generalised eigenvalue methods [22, 23], and resultant based methods [6]. There is, however, still much work in terms of parameterising the original problem, as different paremeterisations may yield completely different results [25]. To produce solvers for the four systems of equations in (5), (8), (9), (11) we have chosen to use the automatic generator [25]. The resulting code (in Matlab or C), typically involve five steps: (i) calculation of the coefficients of the polynomial equations from input data; (ii) generation of the elimination matrix; (iii) calculation of the eigenvalue matrix; (iv) calculation of eigenvalues and eigenvectors; and (v) extraction of the solutions from the eigenvectors and eigenvalues.

However, in some of the cases the automatically generated solvers end up being both large and slow. The reason is that the automatic method of generating the code for step (i) – the calculation of the coefficients of the polynomial equations – becomes large. Therefore, we have hand-coded this calculation. The resulting code is significantly smaller and faster, which is shown in Table 1 (the table is further explained in Section 6.1). These experiments have been run in Matlab. Note that in the table subscript index 1 denotes the automatically generated solvers and 2 our suggested solvers.

It turns out that the problems have, in the presented order, 8, 64, 4 and 32 solutions, respectively, see Table 1. After solving for the modification factors b_i and $\bar{b}_i$ the point pairs can be modified accordingly. After this some known three or four point method for rigid registration can be used to achieve a rotation $\mathbf{R}$, a translation $\mathbf{t}$ and a scale s if needed. This aligns the modified triplets perfectly.

4 Minimal Solvers in RANSAC

Given two point clouds $\mathcal{P}$ and $\mathcal{Q}$ and a set of tentative point matches $(\mathbf{p}_i, \mathbf{q}_i)$, we suggest the use of our presented minimal solvers in a RANSAC framework [14] to find a large inlier set. First, use Algorithm 1 on both $\mathcal{P}$ and $\mathcal{Q}$ separately. If the size of the point cloud is small RANSAC can be exchanged for an exhaustive search. Then, choose one of the solvers from Sections 3.2, 3.2, 3.3, 3.3 and use Algorithm 2 on the two point clouds. In both cases, the number of iterations (or stopping criteria) and a tolerance has to be chosen in order to define the inliers and count the consensus set.

Algorithm 1: RANSAC refinement within a map

```
1 for a given number of iterations do
2     Randomly select a one point pair $(\mathbf{p}_i, \mathbf{p}_j) \in \mathcal{P}$.
3     Use the one point solver on the pair.
4     Transform all of $\mathcal{P}$ accordingly using $\mathbf{B}_i$ and $b_i$.
5     Count the consensus set and keep the solution if it is the best so far.
6 end
```

5 Finding Pair Matches

In our experiments we describe a 3D point using the mode of the ORB descriptors [35] for the corresponding image points. Given two point clouds $\mathcal{P}$ and $\mathcal{Q}$ with points $\mathbf{p}_k$ and $\mathbf{q}_l$, respectively, and the Hamming distance $d_H(\mathbf{p}_k, \mathbf{q}_l)$ we say that two points are a tentative match if

$$\begin{cases} \operatorname{argmin}_j d_H(\mathbf{p}_k, \mathbf{q}_j) = l, \\ \operatorname{argmin}_i d_H(\mathbf{p}_i, \mathbf{q}_l) = k. \end{cases} \tag{12}$$

Algorithm 2: RANSAC for m point solver

```
1  for a given number of iterations do
2      Randomly select $m$ data point pairs $(\mathbf{p}_i, \mathbf{q}_i)$.
3      Use the right $m$ point solver for the chosen pairs.
4      for all solutions do
5          Transform the $m$ point pairs using $\mathbf{B}_i$, $\bar{\mathbf{B}}_i$, $b_i$ and $\bar{b}_i$.
6          Do rigid registration for the modified point pairs to get $\mathbf{R}$, $\mathbf{t}$ (and $s$).
7          Transform $\mathcal{P}$ and $\mathcal{Q}$ accordingly, using both $\mathbf{B}_i$, $\bar{\mathbf{B}}_i$, $b_i$, $\bar{b}_i$ and $\mathbf{R}$, $\mathbf{t}$ (and
            $s$).
8          Count the consensus set and keep the solution if it is the best so far.
9      end
10 end
```

Note that the same thing can be done within the maps, with the requirement that $k \neq l$. The step of finding tentative matches could potentially be solved using a learning based approach instead [34, 45, 46, 49].

To refine these tentative point pairs we use the suggested algorithms in Section 4. For this we use that the covariances $\mathbf{C} = \sigma^2(\mathbf{J}^T\mathbf{J})^{-1}$ of the point clouds, using the Jacobian $\mathbf{J}$ from SLAM and the estimated noise level σ. Eigenvalue decomposition of $\mathbf{C}$ gives the modes of variation, and the modes for specific points can be found by choosing the right entries.

6 Experimental Evaluation

We have first evaluated the presented solvers in terms of speed and numerical stability and then we used them on both a pair of simulated point clouds and point clouds from a SLAM system run in a real environment. For the latter experiments we have used an even (or as even as possible) division of the modes on the two maps, i.e. with $k = 1$ for the three point solvers and $k = 3$ for the four point solvers. Nevertheless, all different solvers have been developed.

6.1 Evaluation of Minimal Solvers

We first ran experiments where the suggested minimal solvers are compared to the automatically generated solvers in terms of robustness and efficiency.

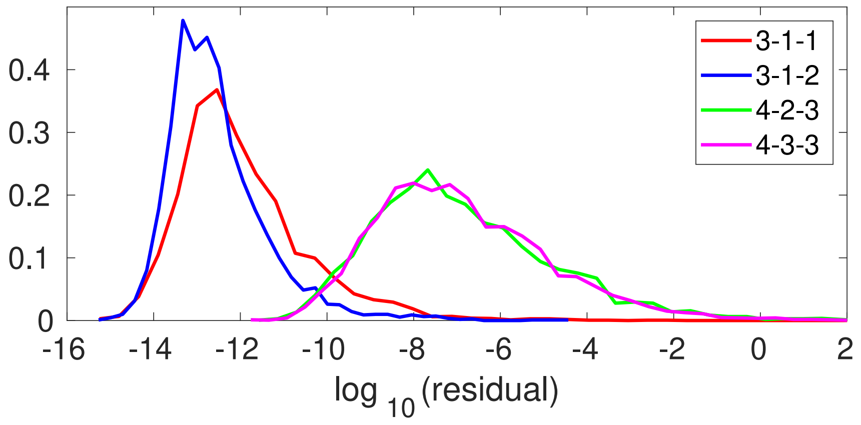


Figure 2: The plot shows histograms over the residuals for the presented minimal solvers.

Robustness and Executions Times

To evaluate the numerical stability and the speed of the solvers we generated a large number of synthetic problem instances for each solver. Then, the equations were evaluated for the found solutions. Figure 2 shows normalised histograms of the maximum equation residual for all solvers – in total 4500 residuals per solver. We denote the *x* point solver with *a* modes added to the first map and *b* modes to the second map *x*-*a*-*b*. Hence, we denote the three point solver with known scale from Section 3.2 by 3-1-2, the one with unknown scale from Section 3.3 by 3-1-1, and correspondingly for the four points solvers, i.e. 4-3-3 and 4-2-3, respectively. The three point solvers are both more numerically robust, which is expected since they use less data and have fewer solutions than the four point solvers. However, all the solvers show a good stability. The histograms for other mode divisions look similar.

Furthermore, in Table 1 the template sizes as well as the size of the solver file and median executions times in Matlab are presented. For comparison, we have included both the values for the automatically generated solvers – denoted by subscript 1 – and the suggested, improved solvers – denoted by subscript 2. From the table it is clear that we decrease the size of the solver file in all cases and that the execution times are reduced significantly for the solves with unknown scale compared to the automatically generated solvers. Note that the numerical stability, displayed in Figure 2, is the same for both versions of the solvers.

6.2 Using the Minimal Solvers on Maps

This section contains three experiments: one where the found inliers are investigated for a real dataset and two experiments – on simulated and real data – where the matches are used

Table 1: Execution times and eliminations template sizes for the minimal solvers. Subscript 1 shows that the solver was automatically generated and 2 the improved version.

Solver type	Number solutions	Template size	Size [kb]	Execution time [ms]
3-1-1_1	4	9×13	197	0.83
3-1-1_2	4	9×13	2.1	0.24
3-1-2_1	8	18×26	8.2	0.21
3-1-2_2	8	18×26	2.9	0.29
4-2-3_1	32	267×299	10 400	77 000
4-2-3_2	32	267×299	165	2.3
4-3-3_1	64	817×881	192	21
4-3-3_2	64	817×881	160	21

for map merging. For the experiments on real data we have generated 3D reconstructions using a SLAM system built on ORB features [35] and IMU data [17], where the matches are filtered using the technique from [42] and the solution is optimised using [20]. Hence, the scale of the maps is already known, and we will be using our solvers from Section 3.2 for these experiments. The tentative matches were found according to Section 5.

Inlier Detection for Real Data

To test the inlier detection we conducted an experiment on three reconstructed 3D maps, see Figure 3. They capture a room, where the first map has a loop closing problem – one of the revisited walls has been doubled – and the ratio of the maps are slightly different.

The tentative matches between the maps were found as described in Section 5. Thereafter, we estimated matches using Algorithm 2 with the 3-1-2 solver (Our 3 pt); Algorithm 2 with the 4-3-3 solver (Our 4 pt); three point Procrustes in a RANSAC loop (Pro 3 pt); and four point Procrustes in a RANSAC loop (Pro 4 pt). All solvers were embedded in the same RANSAC loop in order for them to run on the same randomly selected points (either three or four points). The number of iterations were fixed at 2000 and the solvers were evaluated for different thresholds between 1 to 15 cm. For each of these thresholds the number of correctly and erroneously detected matches were counted by comparing to manual annotations of the tentative matches. There were in total 290 tentative matches between map 1 and 2, 277 between map 1 and 3, and 278 between map 2 and 3. Of these 845 matches 525 were true matches and 320 were false.

The true positive rates (TPR) and the false positive rates (FPR) for the different solvers were calculated. The averages over ten such runs have been plotted as ROC curves in Figure 4. If enough true positives are found, these can be used to close the loop. Our four point solver shows promising results concerning this, as it gives a higher TPR for parts of the low FPR values.

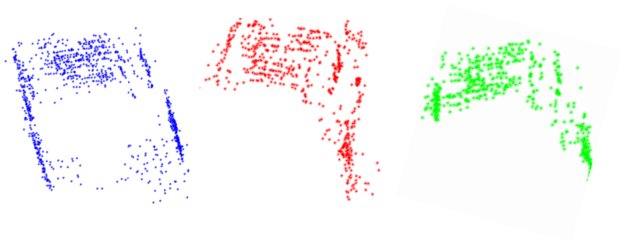


Figure 3: Three local maps that were used to evaluate the minimal solvers in terms of inlier detection. The true labels were annotated manually.

However, as the results are more unclear for the lowest FPR, we wanted to test the found inliers, to know if the false positives that were found by the methods would bring any problems, or if they were just false positives that happened to be close in 3D. Therefore, we used the found matches in a merging algorithm that allows for statistical deformations [16][2]. The results can be seen in Figure 5. Notice that the left wall is doubled in the left map, where Procrustes has been used, while our method solves this. This indicates the usefulness of our methods.

Map Merging on Simulated Data

Furthermore, we simulated a similar setup – two mappings capturing half of a room of size $3 \times 6 \times 2$ m each, see the blue and red maps, respectively, in Figure 6. For this setup it is important not only how many true positives that are found, but which. The point clouds were reconstructed using [52] and as tentative matches both true and random false matches were added. In this case the scales of the point clouds are unknown. The results after match refinement using four point Procrustes and our 4-2-3 solver is shown in Figure 6. Our method finds enough matches to close the loop. After aligning the merged maps to the ground truth map the mean Euclidean error for the four point Procrustes map was 0.43, while is was 0.30 for our four point method. The maximum Euclidean error over all map points was 1.23 and 0.60, respectively. We ran the same experiment with some different noise levels and thresholds, using 30 iterations for each setup, and counted the number of times that matches were detected on both sides of the room. The rate for the different methods are shown in Table 2. Our four point solver outperforms the others and our three point solver is better than Procrustes for low noise levels, while the results indicate that two modes is not enough to capture the uncertainty for higher noise levels.

[2]`https://github.com/gabrielleflood/mapmerging/`

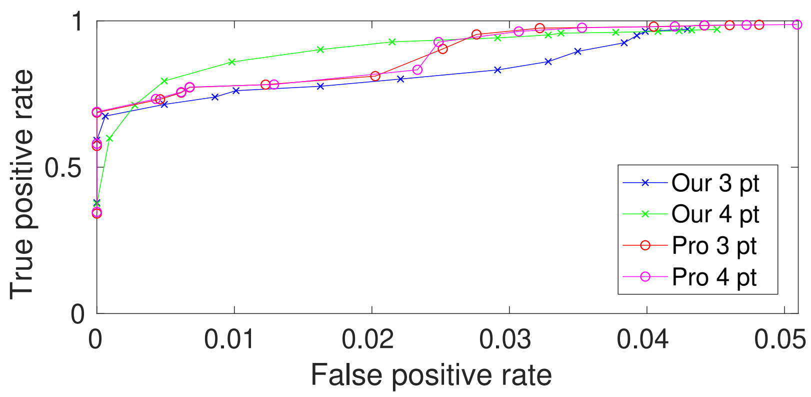


Figure 4: The plot shows the ROC curve for the TPR and FPR w.r.t. inlier threshold for our suggested solvers and Procrustes registration. The algorithms were run on a total of 845 tentative matches.

7 Conclusions

In this paper we have presented several new minimal solvers for matching of 3D point clouds, both with known and unknown scale, together with a framework for how to use the solvers in a RANSAC loop. We have also suggested to use a similar method within maps to eliminate duplicates and solve for loop closing problems. The solvers were evaluated both in terms of efficiency and performance and show better results than the widely used Procrustes method. The output from the suggested system can, e.g. be combined with a map merging system that allows for statistical deformations. This system captures matches that are more spread in the point cloud compared to fully rigid registration and can therefore be used to improve the global map compared to the local map.

Acknowledgments

This work was partially supported by the strategic research projects ELLIIT and eSSENCE, the Swedish Foundation for Strategic Research project, Semantic Mapping and Visual Navigation for Smart Robots (grant no. RIT15-0038) and Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation.

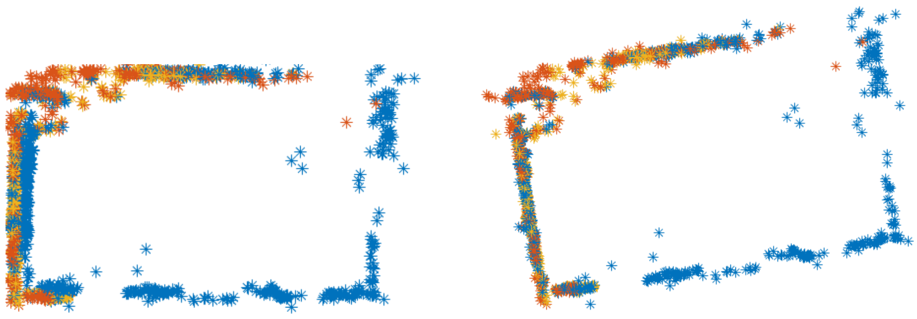


Figure 5: The merged room maps from Figure 3. To the left, matches have been found using three point Procrustes and to the right using our 3-1-2 solver.

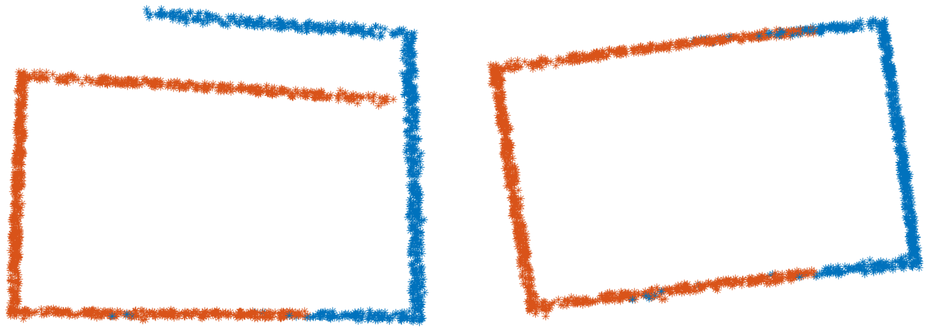


Figure 6: The results after merging the simulated maps using matches from four point Procrustes to the left and our four point method (4-2-3) to the right.

Table 2: The proportion of times that correct inlier matches were found on both sides of the room.

	$\sigma = 0.001$			$\sigma = 0.005$		
Threshold	0.15	0.20	0.25	0.15	0.20	0.25
Our 3 pt	0.13	0.23	0.43	0.07	0.20	0.40
Our 4 pt	0.67	0.77	0.83	0.70	0.73	0.87
Pro 3 pt	0	0.03	0.20	0.07	0.20	0.27
Pro 4 pt	0.03	0.03	0.17	0.07	0.20	0.27

References

[1] Dror Aiger, Niloy J Mitra, and Daniel Cohen-Or. 4-points congruent sets for robust pairwise surface registration. In *ACM SIGGRAPH 2008 papers*, pages 1–10, 2008.

[2] Fouad Amer and Mani Golparvar-Fard. Crowd-sourced visual data collection for monitoring indoor construction in 3d. In *The Second European and Mediterranean Structural Engineering and Construction Conference*. ISEC, International Structural Engineering and Construction Society, 2018.

[3] Ilze Andersone. Heterogeneous map merging: State of the art. *Robotics*, 8(3):74, 2019.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. SURF: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[5] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor fusion IV: control paradigms and data structures*, volume 1611, pages 586–606. International Society for Optics and Photonics, 1992.

[6] Snehal Bhayani, Zuzana Kukelova, and Janne Heikkila. A sparse resultant based method for efficient minimal solvers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1770–1779, 2020.

[7] Andrey Bokovoy, Kirill Muraviev, and Konstantin Yakovlev. Map-merging algorithms for visual slam: Feasibility study and empirical evaluation. In *Russian Conference on Artificial Intelligence*, pages 46–60. Springer, 2020.

[8] Fred L Bookstein. *Morphometric tools for landmark data: geometry and biology*. Cambridge University Press, 1997.

[9] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. BRIEF: Binary robust independent elementary features. In *European conference on computer vision*, pages 778–792. Springer, 2010.

[10] Yang Chen and Gérard Medioni. Object modelling by registration of multiple range images. *Image and vision computing*, 10(3):145–155, 1992.

[11] David A Cox, John Little, and Donal O'shea. *Using algebraic geometry*, volume 185. Springer Science & Business Media, 2006.

[12] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 29(6), 2007.

[13] H Durrant-Whyte and T. Bailey. Simultaneous localization and mapping (slam): Part i the essential algorithms. *Robotics and Automation Magazine*, 13(2), 2006.

[14] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[15] Gabrielle Flood, David Gillsjö, Anders Heyden, and Kalle Åström. Efficient merging of maps and detection of changes. In *Proc. Scandinavian Conference on Image Analysis (SCIA)*. Springer, 2019.

[16] Gabrielle Flood, David Gillsjö, Patrik Persson, Anders Heyden, and Kalle Åström. Generic merging of structure from motion maps with a low memory footprint. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4385–4392. IEEE, 2021.

[17] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Technical Report GT-IRIM-CP & R-2015-001, Georgia Institute of Technology, 2015.

[18] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018.

[19] Daniel F Huber and Martial Hebert. Fully automatic registration of multiple 3d data sets. *Image and Vision Computing*, 21(7):637–650, 2003.

[20] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *International Journal of Robotics Research*, 31(2), 2012.

[21] David G Kendall. A survey of the statistical theory of shape. *Statistical Science*, pages 87–99, 1989.

[22] Z. Kukelova, M. Bujnak, and T. Pajdla. Polynomial eigenvalue solutions to the 5-pt and 6-pt relative pose problems. In *Proceedings of the British Machine Vision Conference*, pages 56.1–56.10. BMVA Press, 2008. doi:10.5244/C.22.56.

[23] Zuzana Kukelova, Martin Bujnak, and Tomas Pajdla. Polynomial eigenvalue solutions to minimal problems in computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1381–1393, 2011.

[24] Martin Larsson, Kalle Åström, and Magnus Oskarsson. Registration and merging maps with uncertainties. In *2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, pages 206–212. IEEE, 2018.

[25] Viktor Larsson, Kalle Åström, and Magnus Oskarsson. Efficient solvers for minimal problems by syzygy-based reduction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 820–829, 2017.

[26] Chien-Chou Lin, Yen-Chou Tai, Jhong-Jin Lee, and Yong-Sheng Chen. A novel point cloud registration using 2d image features. *EURASIP Journal on Advances in Signal Processing*, 2017(1):1–11, 2017.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] Andre Mateus, Srikumar Ramalingam, and Pedro Miraldo. Minimal solvers for 3d scan alignment with pairs of intersecting lines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7234–7244, 2020.

[29] Pedro Miraldo, Surojit Saha, and Srikumar Ramalingam. Minimal solvers for mini-loop closures in 3d multi-scan alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9699–9708, 2019.

[30] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[31] Carl Olsson, Olof Enqvist, and Fredrik Kahl. A polynomial-time bound for matching and registration with outliers. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[32] G Dias Pais, Srikumar Ramalingam, Venu Madhav Govindu, Jacinto C Nascimento, Rama Chellappa, and Pedro Miraldo. 3dregnet: A deep neural network for 3d point registration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7193–7203, 2020.

[33] Oliver Pink. Visual map matching and localization using a global feature map. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–7. IEEE, 2008.

[34] Tobias Plötz and Stefan Roth. Neural nearest neighbors networks. *Advances in Neural Information Processing Systems*, 31:1087–1098, 2018.

[35] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *2011 International conference on computer vision*, pages 2564–2571. Ieee, 2011.

[36] Szymon Rusinkiewicz and Marc Levoy. Efficient variants of the icp algorithm. In *Proceedings third international conference on 3-D digital imaging and modeling*, pages 145–152. IEEE, 2001.

[37] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 2018.

[38] Peter H Schönemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[39] George AF Seber. *Multivariate observations*, volume 252. John Wiley & Sons, 2009.

[40] Robert W Sumner, Johannes Schmid, and Mark Pauly. Embedded deformation for shape manipulation. In *ACM SIGGRAPH 2007 papers*, pages 80–es. 2007.

[41] Shiqin Sun, Benlian Xu, Yidan Sun, and Zhen Sun. Sparse pointcloud map fusion of multi-robot system. In *2018 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 270–274. IEEE, 2018.

[42] George Vogiatzis and Carlos Hernández. Video-based, real-time multi-view stereo. *Image and Vision Computing*, 29(7), 2011.

[43] Thomas Whelan, Michael Kaess, John J Leonard, and John McDonald. Deformation-based loop closure for large scale dense rgb-d slam. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 548–555. IEEE, 2013.

[44] Thomas Whelan, Stefan Leutenegger, R Salas-Moreno, Ben Glocker, and Andrew Davison. Elasticfusion: Dense slam without a pose graph. Robotics: Science and Systems, 2015.

[45] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 2020.

[46] Kwang Moo Yi, Eduard Trulls, Yuki Ono, Vincent Lepetit, Mathieu Salzmann, and Pascal Fua. Learning to find good correspondences. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2666–2674, 2018.

[47] Shuien Yu, Chunyun Fu, Amirali K Gostar, and Minghui Hu. A review on map-merging methods for typical map types in multiple-ground-robot slam solutions. *Sensors*, 20(23):6988, 2020.

[48] Chi Yuan, Xiaoqing Yu, and Ziyue Luo. 3d point cloud matching based on principal component analysis and iterative closest point algorithm. In *2016 International Conference on Audio, Language and Image Processing (ICALIP)*, pages 404–408. IEEE, 2016.

[49] Chen Zhao, Zhiguo Cao, Chi Li, Xin Li, and Jiaqi Yang. Nm-net: Mining reliable neighbors for robust feature correspondences. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 215–224, 2019.

[50] Xun S Zhou and Stergios I Roumeliotis. Multi-robot slam with unknown initial correspondence: The robot rendezvous case. In *2006 IEEE/RSJ international conference on intelligent robots and systems*, pages 1785–1792. IEEE, 2006.

[51] Danping Zou and Ping Tan. Coslam: Collaborative visual slam in dynamic environments. *IEEE transactions on pattern analysis and machine intelligence*, 35(2), 2013.

[52] O. Enqvist, F. Kahl, and C. Olsson, "Non-sequential structure from motion," in *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*. IEEE, 2011, pp. 264–271.

[53] R. Hartley, J. Trumpf, Y. Dai, and H. Li, "Rotation averaging," *International journal of computer vision*, vol. 103, no. 3, pp. 267–305, 2013.

[54] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *International workshop on vision algorithms*. Springer, 1999, pp. 298–372.

Printed by Media-Tryck, Lund 2023 NORDIC SWAN ECOLABEL 3041 0903

LUND UNIVERSITY

Doctoral Theses in Mathematical Sciences 2023:2
ISBN 978-91-8039-643-1
ISSN 1404-0034