LUND UNIVERSITY

Mapping and Merging Using Sound and Vision

Automatic Calibration and Map Fusion with Statistical Deformations

Flood, Gabrielle

2021

Document Version:
Publisher's PDF, also known as Version of record

Link to publication

Citation for published version (APA):
Flood, G. (2021). *Mapping and Merging Using Sound and Vision: Automatic Calibration and Map Fusion with Statistical Deformations*. [Doctoral Thesis (compilation), Mathematics (Faculty of Engineering)]. Lund University / Centre for Mathematical Sciences /LTH.

Total number of authors:
1

General rights
Unless other specific re-use rights are stated the following general rights apply:
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.
• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal

Read more about Creative commons licenses: https://creativecommons.org/licenses/

Take down policy
If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

LUND UNIVERSITY

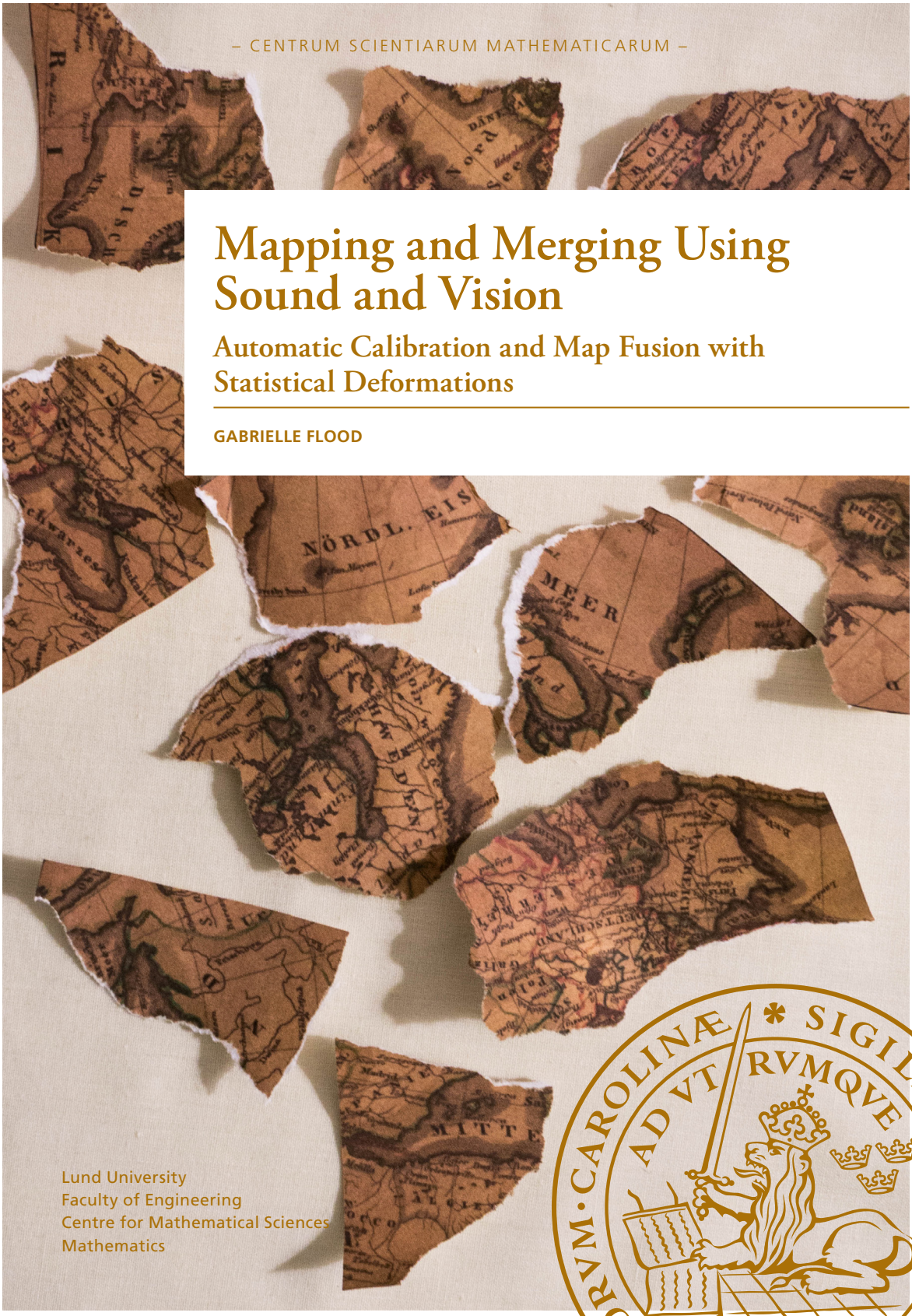
PO Box 117
221 00 Lund
+46 46-222 00 00

– CENTRUM SCIENTIARUM MATHEMATICARUM –

Mapping and Merging Using Sound and Vision

Automatic Calibration and Map Fusion with Statistical Deformations

GABRIELLE FLOOD



Lund University
Faculty of Engineering
Centre for Mathematical Sciences
Mathematics

Mapping and Merging Using Sound and Vision

Automatic Calibration and Map Fusion with Statistical Deformations

by Gabrielle Flood

LUND
UNIVERSITY

Cover illustrations: Printed, cut into pieces, taped and photographed version of "19th Century Map of Europe" photograph by Paul Simpson, 2009-06-15, https://www.flickr.com/photos/paulsimpson1976/3629546523. Licensed under CC BY 2.0; https://creativecommons.org/licenses/by/2.0/legalcode.

pp. i–88	©	Gabrielle Flood, 2021
Papers I, VI	©	Springer Nature Switzerland AG, 2019
Paper II	©	2019 IEEE
Paper III	©	2020 IEEE
Papers IV, V, VII	©	2021 IEEE
Paper VIII	©	Gabrielle Flood, Erik Tegler, David Gillsjö, Anders Heyden and Kalle Åström, 2021

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

Doctoral Theses in Mathematical Sciences 2021:10
ISSN: 1404-0034
ISBN: 978-91-8039-055-2 (print)
ISBN: 978-91-8039-056-9 (electronic)
LUTFMA-1075-2021

Printed in Sweden by Media-Tryck, Lund University, Lund 2021

Media-Tryck is a Nordic Swan Ecolabel certified provider of printed material. Read more about our environmental work at www.mediatryck.lu.se

Printed matter 3041 0903

MADE IN SWEDEN

Those who explore an unknown world are travelers without a map:
the map is the result of the exploration.
The position of their destination is not known to them,
and the direct path that leads to it is not yet made.

Hideki Yukawa, 1986

Abstract

Over the last couple of years both cameras, audio and radio sensors have become cheaper and more common in our everyday lives. Such sensors can be used to create maps of where the sensors are positioned and the appearance of the surroundings. For sound and radio, the process of estimating the sender and receiver positions from time of arrival (TOA) or time-difference of arrival (TDOA) measurements is referred to as automatic calibration. The corresponding process for images is to estimate the camera positions as well as the positions of the objects captured in the images. This is called structure from motion (SfM) or visual simultaneous localisation and mapping (SLAM). In this thesis we present studies on how to create such maps, divided into three parts: to find accurate measurements; robust mapping; and merging of maps.

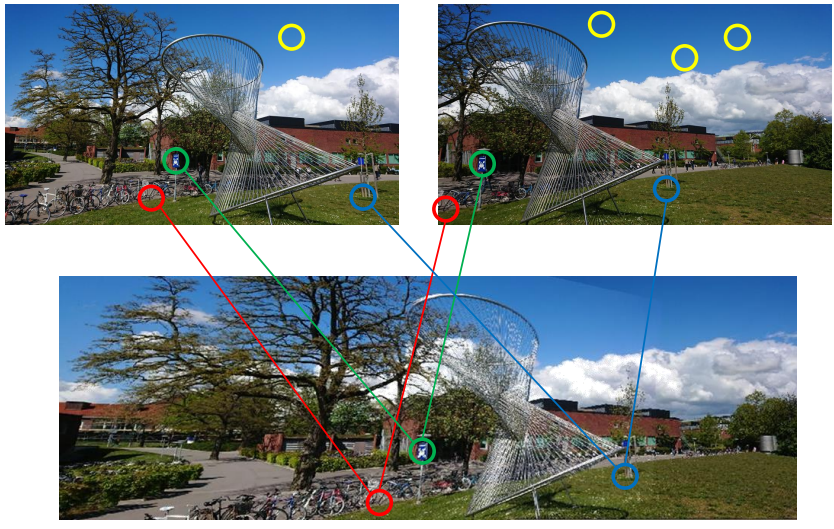
The first part is treated in Paper I and involves finding precise – on a subsample level – TDOA measurements. These types of subsample refinements give a high precision, but are sensitive to noise. We present an explicit expression for the variance of the TDOA estimate and study the impact that noise in the signals has. Exact measurements is an important foundation for creating accurate maps.

The second part of this thesis includes Papers II–V and covers the topic of robust self-calibration using one-dimensional signals, such as sound or radio. We estimate both sender and receiver positions using TOA and TDOA measurements. The estimation process is divided in two parts, where the first is specific for TOA or TDOA and involves solving a relaxed version of the problem. The second step is common for different types of problems and involves an upgrade from the relaxed solution to the sought parameters. In this thesis we present numerically stable minimal solvers for both these steps for some different setups with senders and receivers. We also suggest frameworks for how to use these solvers together with RANSAC to achieve systems that are robust to outliers, noise and missing data. Additionally, in the last paper we focus on extending self-calibration results, especially for the sound source path, which often cannot be fully reconstructed immediately.

The third part of the thesis, Papers VI–VIII, is concerned with the merging of already estimated maps. We mainly focus on maps created from image data, but the methods are applicable to sparse 3D maps coming from different sensor modalities. Merging of maps can be advantageous if there are several map representations of the same environment, or if there is a need for adding new information to an already existing map. We suggest a compact map representation with a small memory footprint, which we then use to fuse maps efficiently. We suggest one method for fusion of maps that are pre-aligned, and one where we additionally estimate the coordinate system. The merging utilises a compact approximation of the residuals and allows for deformations in the original maps. Furthermore, we present minimal solvers for 3D point matching with statistical deformations – which increases the number of inliers when the original maps contain errors.

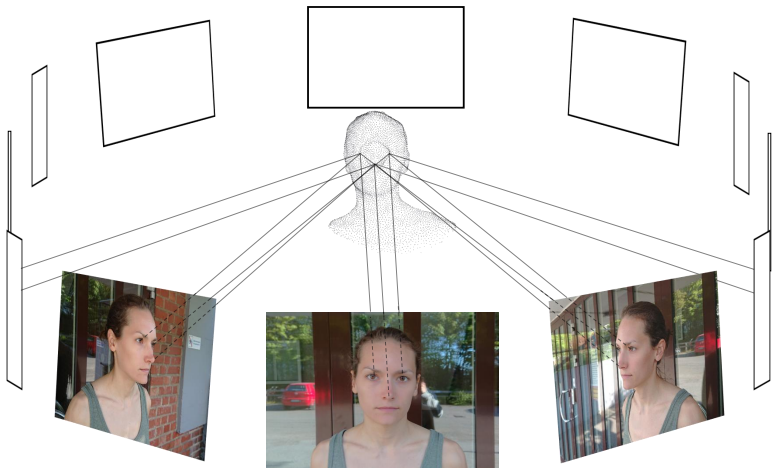
Popular Summary

Can computers understand the world as well as humans do? Can a robot navigate on its own? Can a drone remember the appearance of a room? And can you from a number of images learn what the environment they depict looks like? All of these questions have a connection to the contents of this thesis.



The figure shows how to combine two different images of the same scene into a panoramic image. This requires the identification of interesting and matching points in both images, for example marked by the red, green and blue circles. The yellow circles show an example of points that are bad to use, since these are not unique.

Computer vision is an area which focuses on teaching computers how to gain knowledge and understanding from images, just as humans do when they see something. In many ways cameras are similar to the human eye, and humans and other animals are in general very good at understanding which type of objects that are in front of them or how far away different things are. The idea with computer vision is that computers should be able to do this as well. One thing that images can be used for is to create *3D models* or *maps* of the world. We humans can determine the depth of what we see and using this, our memory and our experiences we can create our own map of, for example, a room or a flat. The same thing can actually be achieved digitally, using computers. The procedure is similar to that of creating panoramic images. Today, most mobile phones have both a camera and an application for creating panoramic images – that is, many small images stitched together to one larger image. The essential parts of image stitching is to find *interesting* points that can be seen in both images and then to move the images such that these points match. An example of this can be seen in the figure above.

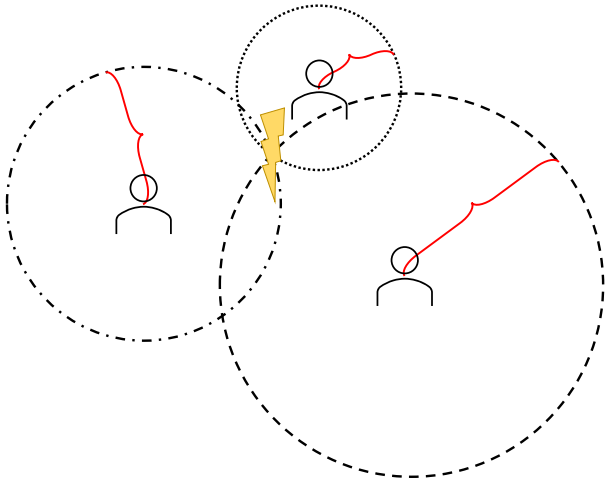


Using several images the 3D position of the "interesting" points can be found. If there are enough images one can create a 3D model, here represented as a point cloud, of the person in the images.

Similarly, with enough images taken from different angles, these can be "stitched" into a 3D model. The depth can be recovered since the images are taken at different positions, as can be seen in the figure above. To estimate 3D coordinates using interesting image points and known camera positions is called *triangulation* and if the camera positions are unknown but estimated as well, we call it *structure from motion*. If sufficiently many points are triangulated a map of the environment is obtained.

Such maps can also be created using other types of sensors and signals, such as microphones and sound. Sound describes some sort of information in one dimension in the same way that images do in two dimensions. Most people have probably, at some point, counted the seconds from the flash of a lightning until the thunderclap can be heard. Through this, you know how far away the thunderstorm is. You do not know in which direction it is located, but you do know that it is positioned somewhere on a circle where you are the centre, and the counted distance is the radius. If you could also know the thunder's distance to two other positions, you could draw two more circles and then the thunderstorm would be located where these intersect. This has been illustrated in the figure on the next page. The same procedure can be used for microphones and loudspeakers that are set up in a room, and if there are sufficiently many it is actually enough to only know the distances between them to compute the relative positions of both the microphones and the speakers. This gives a map of the microphone positions, in many ways similar to the map that can be created from images.

Once the 3D maps are obtained, they can be used for *positioning* – to find out where in the



The image illustrates how you can find out where a thunderstorm is, if three people hear the thunder from different positions. By drawing a circle around each person, the position of the storm is given by the intersection of the circles.

map you are, either using sound or images. One example of a system that uses signals that are similar to sound is GPS, which is used a lot for outdoor positioning and navigation. The GPS does, however, work less well indoors, and because of this it can be good to have other systems that can be used in similar ways for indoor positioning. For the positioning to be as good as possible it is important that the map is as good as possible and in general the maps get better if more measurements are used (for example more images). Therefore, it is a good thing to be able to merge different maps of the same environment, to a more exact map and to be able to update it in case something in the environment changes. This can be done by identifying corresponding interesting points in the different maps and then stitching them so that they coincide – just as for panoramic stitching and triangulation.

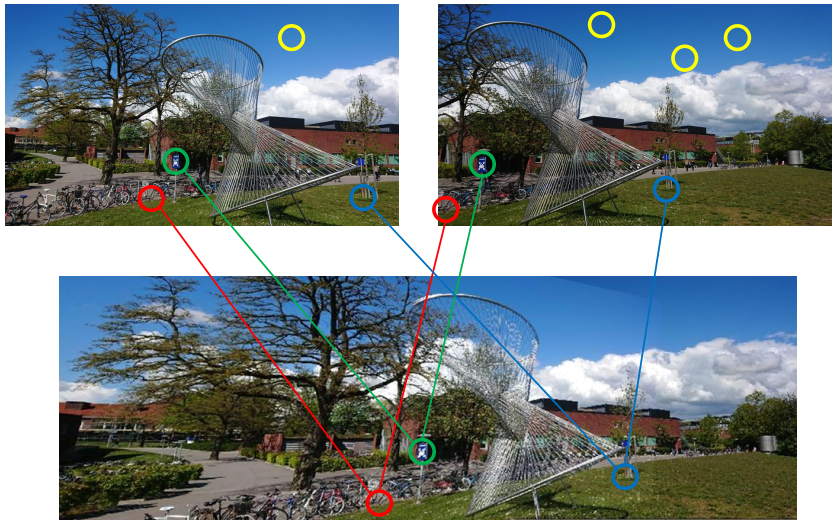
Map merging can, for example, be useful for self-driving cars. Today, many cars have one or several cameras which can be used to determine the position of the car in an already known environment. Imagine that we have a map of a city and that a car is driving through this city. While driving, it will collect many images and using these images it can create its own, *local* map of the city. Now, if the local map can be added to the large, *global* map, the updated global map will after that contain more information than before. This makes it more exact. So if all cars that drive through the city can do the same thing, the map will gradually get better. Also, if some infrastructure of the city is changed, this will also be captured in the map, without the map being re-created.

In this thesis we first focus on finding good measurements between microphones and loudspeakers. We then cover the topic of creating maps using such measurements and the more exact the measurements are, the better will the resulting maps be. Finally, we show how

several such maps – consisting of 3D point clouds – can be fused into a single, more accurate map. The local maps can be created either using sound or images, as in the examples above.

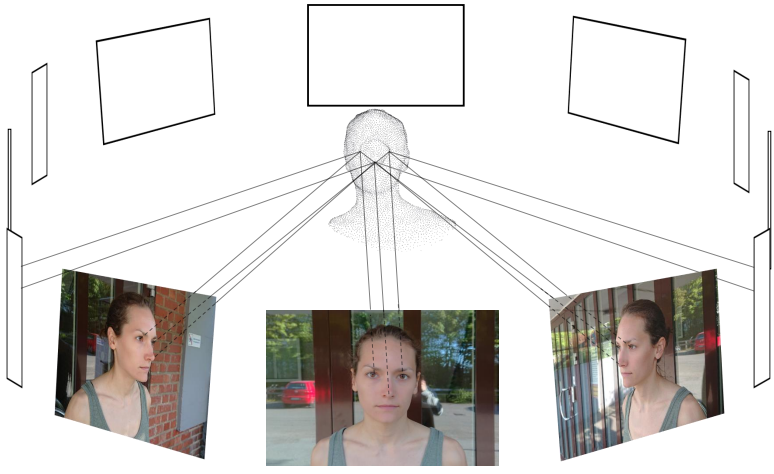
Populärvetenskaplig sammanfattning

Kan datorer förstå världen lika bra som människor? Kan en robot navigera på egen hand? Kan en drönare komma ihåg hur ett rum ser ut? Och kan man från platta bilder förstå hur omgivningen de avbildar ser ut? Allt detta är frågor med koppling till innehållet i denna avhandling.



Bilden visar sammansättning av två olika bilder som fångar samma scen till en panoramabild. För detta krävs att man identifierar intressanta och matchande punkter i de båda bilderna, exempelvis de som är markerade av de röda, gröna och blå cirklarna. De gula cirklarna markerar punkter som är dåliga att använda, eftersom dessa inte är unika.

Datorseende är ett område som handlar om att lära datorer att förstå och utläsa information ur digitala bilder, precis som vi människor gör när vi ser någonting. Det mänskliga ögat är i många avseenden likt en kamera och generellt är människor och djur väldigt bra på att förstå saker – som vad det är man ser eller hur långt bort olika saker är. Datorseende handlar om att lära datorer att göra samma sak. En sak som bilder kan användas till är att skapa *3D-modeller* eller *kartor* av verkligheten. Vi människor kan bedöma djupet i det vi ser, och med hjälp av detta, vårt minne och våra erfarenheter kan vi skapa oss våra egna kartor av hur exempelvis ett rum eller en lägenhet ser ut. Samma sak kan man göra digitalt, med hjälp av datorer. Tillvägagångssättet är ganska likt det man använder när man skapar panoramabilder. I dag har de flesta mobiltelefoner både en kamera och en funktion för att skapa panoramabilder – det vill säga stora bilder som egentligen är flera ihopklistrade bilder. Om man vill klistra ihop två bilder gör man väsentligen så att man noterar *intressanta* punkter som syns i båda bilderna och ser till att dessa matchar. Ett exempel på detta syns i figuren ovan.

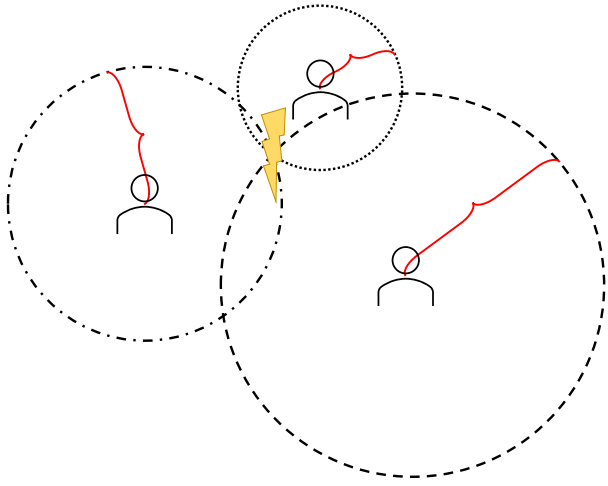


Med hjälp av flera bilder bestäms positionen i 3D för de "intressanta" punkterna. Om man har tillräckligt många bilder kan man skapa en 3D-modell, här i form av ett moln av punkter.

På samma sätt kan man, om man har tillräckligt många bilder från olika vinklar, "klistra ihop" dem till en 3D-modell. Djupet fås, precis som när människor ser, av att bilderna är tagna från olika ställen, se bilden ovan. Att med hjälp av kamerapositioner och intressanta punkter i bilder beräknar positionen för 3D-punkter kallas det för *triangulering* och om man dessutom samtidigt hittar kamerapositionerna brukar det kallas *struktur och rörelse*, eller *structure from motion*. När tillräckligt många punkter har triangulerats har man en typ av karta av omgivningen.

Sådana här kartor kan även skapas med hjälp av andra sensorer och signaler, till exempel mikrofoner och ljud. Precis som att en bild beskriver någon form av information i två dimensioner så gör ljudet det i en dimension. De allra flesta har nog vid något tillfälle räknat sekunderna från det att man ser en blixt tills dess att man hör åskknallen. Därigenom vet man hur långt bort åskan är. Man vet inte vilket håll den kommer ifrån, men man vet att den befinner sig någonstans på en cirkel där man själv är centrum och det uträknade avståndet är radien. Om man dessutom känner till vad avståndet till några fler punkter är kan man rita upp fler cirklar och där dessa skär varandra befinner sig åskan för stunden. Detta finns illustrerat i figuren på nästa sida. Samma sak kan man göra med vanliga mikrofoner och högtalare som står uppställda i ett rum. Finns det tillräckligt många så räcker det att man vet de respektive avstånden för att kunna beräkna de relativa positionerna både för mikrofonerna och för högtalarna. Därigenom får man en karta över mikrofonpositionerna, som på många sätt är lik den karta man kan skapa med hjälp av bilder.

När man väl har 3D-kartor kan dessa användas för *positionering*, alltså för att ta reda på var



Bilden illustrerar hur man kan lista ut var exempelvis ett åskoväder befinner sig, om man är tre personer som hör åskan från olika platser. Genom att rita cirklar runt de olika personerna med radier som motsvaras av avståndet, så vet man att åskan är där cirklarna skär varandra.

man befinner sig, antingen med hjälp av ljud eller bilder. Ett exempel på ett system som använder signaler som liknar ljud är GPS, som används flitigt för utomhuspositionering och navigation. Inomhus fungerar dock GPS sämre, och därför kan det vara bra att ha andra system som kan användas på liknande sätt men för inomhuspositionering. För att positioneringen ska bli så exakt som möjligt är det viktigt att kartan är så exakt som möjligt och kartorna blir generellt bättre ju fler mätningar man gör (exempelvis ju fler bilder man använder). Därför är det bra om man kan slå samman flera olika kartor av samma miljö, för att öka noggrannheten och för att kunna uppdatera kartan om någonting ändras. Detta kan man göra genom att – precis som i bilderna – identifiera motsvarande intressanta punkter i de olika kartorna och pussla ihop dem så att de överlappar.

Ett exempel på när kartsammanslagning kan vara användbart är för självkörande bilar. Många bilar har i dag kameror och dessa kan användas för att bestämma positionen för bilen i en sedan tidigare känd miljö. Vi tänker oss att vi har en karta över en stad och att en bil sedan kör igenom denna stad. Medan detta händer kommer bilen samla in en massa bilder och den kan då skapa sig sin egen, *lokala* karta av de delar av staden som den passerar. Om denna lokala karta sedan kan läggas till den stora, *globala* kartan, så kommer den därefter att innehålla mer information och därmed vara med exakt. Om alla bilar som kör genom staden kan göra detta kommer kartan successivt att bli bättre och om någon infrastruktur i staden ändras kommer detta att återspeglas i kartan utan att den helt behöver göras om.

I den här avhandlingen fokuserar vi först på att hitta exakta mätningar för avstånden mellan mikrofoner och högtalare. Sedan arbetar vi vidare med att skapa kartor med hjälp av sådana

mätningar, och ju mer exakta mätningarna är, desto bättre kommer de slutgiltiga kartorna att vara. Slutligen, om vi har flera sådana kartor – som består av 3D-punktmoln – så visar vi hur dessa kan slås samman till en, mer exakt karta. De lokala kartorna kan vara skapade antingen med hjälp av ljudmätningar eller bilder, som i exemplen ovan.

List of Publications

This thesis is based on the following publications, referred to by their Roman 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.

Main papers

I **Stochastic Analysis of Time-Difference and Doppler Estimates for Audio Signals**
G. Flood, A. Heyden and K. Åström
Pattern Recognition Applications and Methods, Springer International Publishing, Cham, 2019.

Author's contributions: The paper evolved from a previous paper written by KÅ and AH and a conference paper (subsidiary) with the same authors. GF contributed to the theory development and discussions about the paper. GF wrote most of the code for the experiments, together with KÅ, and generated the final results. GF also did a large part of the writing, with input from the co-authors.

II **Robust Self-Calibration of Constant Offset Time-Difference-of-Arrival**
K. Batstone, G. Flood, T. Beleyur, V. Larsson, H. R. Goerlitz, M. Oskarsson and K. Åström
IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019.

Author's contributions: The original ideas were suggested by KÅ, but MO, KB and VL helped to develop them for the paper. The coding was mainly conducted by KÅ, MO, KB and VL, and VL developed the fast solver. GF contributed to the experiments, both in the collection of real data and parts of the synthetic experiments. GF also took part in the writing process together with all the other authors.

III **Upgrade Methods for Stratified Sensor Network Self-Calibration**
M. Larsson, G. Flood, M. Oskarsson and K. Åström
IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2020.

Author's contributions: The ideas were suggested by KÅ, MO and ML, but all authors took part in discussing them. GF helped with the experiments and the writing process.

IV **Fast and Robust Stratified Self-Calibration Using Time-Difference-of-Arrival Measurements**
M. Larsson, G. Flood, M. Oskarsson and K. Åström
IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2021.

Author's contributions: The paper was a natural continuation from Paper III. The ideas were suggested by KÅ, MO and ML, but all authors took part in discussing them. GF contributed to the experiments and the writing process.

V **Extension of Time-Difference-of-Arrival Self Calibration Solutions Using Robust Multilateration**
K. Åström, M. Larsson, G. Flood and M. Oskarsson
29th European Signal Processing Conference (EUSIPCO), 2021.

Author's contributions: The ideas were discussed during the process of earlier papers and suggested by KÅ. All authors took part in developing the ideas, the experiments and the writing.

VI **Efficient Merging of Maps and Detection of Changes**
G. Flood, D. Gillsjö, A. Heyden and K. Åström
Scandinavian Conference on Image Analysis (SCIA), 2019.

Author's contributions: The idea for the merging algorithm came from KÅ, but GF contributed to the development of the original idea to the theory and methods used in the paper. The majority of the code was written by GF, DG and KÅ. GF did most of the writing of the paper, but all the authors contributed, and wrote some parts each.

VII **Generic Merging of Structure from Motion Maps with a Low Memory Footprint**
G. Flood, D. Gillsjö, P. Persson, A. Heyden and K. Åström
25th International Conference on Pattern Recognition (ICPR), 2021.

Author's contributions: The paper was a continuation and generalisation of Paper VI. The ideas were discussed between all authors but KÅ and GF had a leading role in developing them. GF, KÅ and DG wrote most of the code and PP helped in conducting the experiments. GF wrote most of the paper but with help from all authors.

VIII **Minimal Solvers for 3D Map Matching with Statistical Deformations**
G. Flood, E. Tegler, D. Gillsjö, A. Heyden and K. Åström
Manuscript.

Author's contributions: The idea to use minimal solvers to refine matches came from KÅ, but was based on discussions about and work with Paper VII. GF had a leading role in the development of the research and wrote most of the code, with help from KÅ, DG and ET.

Subsidiary papers

Estimating Uncertainty in Time-difference and Doppler Estimates
G. Flood, A. Heyden and K. Åström
7th International Conference on Pattern Recognition Applications and Methods (ICPRAM), 2018.

Paper I is an extended version of this paper.

Acknowledgements

These acknowledgments only cover a few pages of this thesis, but they include people that have been as fundamental for my development and the resulting research as any knowledge, research paper or book has been. There are so many people that have helped me in different ways throughout these years and I would especially like to express my gratitude to:

My thesis advisors, *Kalle Åström*, *Anders Heyden* and *Carl Olsson*. Thank you for guiding me into and through the jungle of academia, helping me prepare so that maybe one day I can explore it on my own. *Kalle*, thank you for everything. For academic expertise and endless number of research ideas. For being supportive and encouraging when I have been struggling. Thank you for not only showing me research but doing it with me and for being a great role-model. I do not think that there is a type of thesis advisor that is good for everyone, but I am certain that you have been the best advisor for me. *Anders*, thank you for all the advice and support and for being down-to-earth when I have been stressed. *Carl*, thank you for great proof-reading of my licentiate and doctoral theses, but also for being an inspiration.

My fellow PhD students at the Centre for Mathematical Sciences, for making hard times easier and good times even better. In particular: *Maria Priisalu*, with whom I shared office for most of my PhD studies. Thank you for acting as my common sense when I have lost it. Thank you for always making me in a better mood, even during the lousiest of mornings, and for laughing both with me and at me (in a friendly way). *Ida Arvidsson*, who has been so much during these five years – a supportive colleague, a fantastic study partner and a close friend all at once. Thank you for letting me go through these years and grow together with you. Thank you for all your help with this thesis and for always listening to my complaints and concerns. I have learnt a lot from you. *Anna Gummeson* and *Linn Öström*, for making Friday afternoons during the pandemic so much better and for proof-reading this thesis. Thank you *Anna*, for being the first person that I actually discussed ongoing research with – during our Master's theses – and for then coming back to MC to become a dear collegue. Thank you *Linn* for being refreshing and cheerful from your first day at MC and becoming a friend in just a few weeks. *Mats Bylund*, for making coffee breaks during the pandemic enjoyable. Thank you for lunch training sessions, for explaining mathematics to me and for always having a reasonable view on things. *David Gillsjö*, for never making me feel stupid, independently of how trivial questions I ask and for great collaboration and discussions throughout the years. *Johan Fredriksson*, for early showing me how cool a PhD student in applied mathematics can be and for answering all my stupid questions concerning life as a PhD student in the beginning of my studies.

My other colleagues. Thank you for being an inspiration, for teaching me about academia and for making coffee breaks fun and interesting. I would particularly like to thank: *Niels*

Christian Overgaard, for good advice and hard questions during my licentiate defense and for proofreading this thesis. Also, for always being open minded and for being the first senior colleague – except for my advisors – that made me feel like a peer. *The co-authors of the papers*, for the collaborations, research discussions and everything you have taught me. *Everyone in the SSF Smart Systems project*, for fruitful discussions and worthwhile presentations that have broadened my view and knowledge. *Eva-Lena Borgström* and *Lena Lööf*, for always being helpful. *Everyone who has trained with me*, once or more often, during the lunch breaks. Thank you for helping me return to work in the afternoon with a slightly clearer mind. I would also like to thank the *Math Building*, for being my second home for the last ten and a half years.

Others that have been of importance for my development as a PhD student, despite not necessarily working at MC. In particular: *Lea Versbach* and *Marcus Klasson* – The Da Vinci PhDs – for many hours spent together during our Master's theses, and an environment that was encouraging and made me feel that working in academia was as good of an option as working in the industry. Also, thank you for great chats during the last five years and for sharing the PhD experience, recognising what I have been going through and sometimes making me feel less weird. *Sara Månsson*, for discussions on PhD related matters, acknowledging both good and bad parts, and for sharing a lot of information and advice on thesis writing.

People outside the academia that are close to me, for joy and recovery outside working hours. My wonderful friends *Christina Rönngren* and *Matilda Hjälle*, for always being completely unconditional. Thank you for your understanding during hectic periods, for supporting me but also for telling me when things have gone too far. Thank you for all the happy and relaxing moments that have been so important in between the studies. *My great friends in the cycling club Lunedi*, for helping me to take my mind off work. Thank you for soothing Sunday fika rides and for forcing me to think about nothing other than lactic acid and to not get dropped every Tuesday and Thursday evening.

Everyone that has been of importance for my way to the PhD studies and still are. Especially: *My family*. Thank you *Mamma* for teaching me the importance of studies, for helping me with literately every homework until I was 18 years old and for arising my interest in pedagogics. Thank you *Pappa* for helping me find my interest in natural sciences – through both nice discussions, Kunskapskanalen and Illustererad vetenskap – and for showing me the beauty of logic. Thank you *Syster* for always taking care of me and for leading the path and sharing with me both the ups and the downs of pursuing a PhD. Thank you for being both just like me and completely different at the same time and for being a great friend. My almost-family childhood friends *Malin Hermansson* and *Veronica Persson-Lilja*, for not giving up on me despite that I moved away and sometimes forget to call. Thank you for always supporting me, for embracing our differences and for asking me about what I am actually doing down here in Skåne time and again. *Johan Andersson*;

there are not words enough to express the gratitude I am feeling. Thank you for helping me through the downs of this journey – because there has indeed been some. Thank you for cooking dinner, for fixing my bike and all those other things that I have down prioritised during hectic periods. Thank you for sometimes questioning me and my decisions, but also respecting them. And last but not least, thank you for listening to every thought about everything and just always being there for me.

Funding

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

Contents

Chapter 1

Introduction

Computer vision refers to a research field where computers are to extract relevant information from images, in a similar way that humans can do. One example of this could be the ability to recognise certain objects in an image, or to read the numbers and letters of a license plate. An image is a two-dimensional representation of the three-dimensional world. To go in the opposite direction, to recover 3D information from images, is called *geometric computer vision*, as it uses the geometry of the camera and scene to regain this information [88]. Given only one image, and no additional information, it is not possible to determine at which distance, or depth, an object is positioned using only geometry. However, if the object is also seen from another image which is taken at another location, the relative depth can be estimated. Using several images to compute both the positions of 3D objects and the cameras that took the photos is known as *structure from motion* (SfM) [35, 59, 93]. Using structure from motion we can create 3D reconstructions – or models – of captured objects. If the reconstruction covers a whole scene, we can refer to it as a map. The distances between the image points and the reprojections of their reconstructions are called measurement errors, see Figure 1.1. If these are zero mean and Gaussian, the maximum likelihood estimate of the 3D reconstruction is obtained by minimising the sum of squared errors. The iterative process that is used for finding these estimates is called *bundle adjustment*. The name refers to the bundle of rays that goes from each 3D point in space to each camera and in general it can be seen as a large, sparse geometric parameter estimation problem [92].

The process of estimating a map and computing the position of cameras can also be referred to as visual *simultaneous localisation and mapping* (SLAM) [23, 27]. Classically, visual SLAM has focused more on the motion, while SfM has been more focused on the structure. Also, visual SLAM often requires there there is only one moving camera and it is often used for online applications, while SfM can be used for unsorted images from differ-

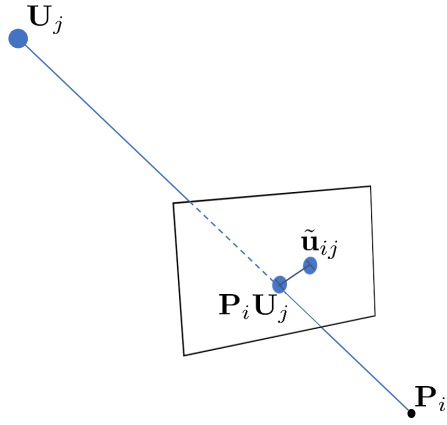


Figure 1.1: The image illustrates the reprojection error. The measured image point is $\tilde{\mathbf{u}}_{ij}$. However, once the 3D point $\mathbf{U}_j$ and camera $\mathbf{P}_i$ are estimated, the projection – which is achieved by intersecting the line from $\mathbf{P}_i$ to $\mathbf{U}_j$ with the image plane – ends up on a slightly different position. The distance between the original image point and the projection $\mathbf{P}_i\mathbf{U}_j$ is called the reprojection error and in order to improve the 3D reconstruction the aim is to make this as small as possible.

ent cameras and commonly works on all images. Nevertheless, the two methods essentially solve the same problem, but are originating from different research fields. However, the term SLAM is also used for other types of sensors, such as lasers, lidar, wifi, etc. In this thesis, we have – in addition to images – also worked with one-dimensional signals such as sound and ultra wide-band (UWB) signals. The same principles that we use are, however, applicable to any similar one-dimensional signal. We will mainly refer to the simultaneous estimation of sender and receiver positions using these signals as *self-calibration* or *automatic calibration*, but again, this is mainly a matter or terminology [22, 47, 71, 108]. Because of its similarities to SfM the process can also be called *structure from sound* when sound signals are used [91]. These calculations require either *time of arrival* (TOA) or *time-difference of arrival* (TDOA) measurements, that is, measurements of the travel time from each sender to each receiver, or the differences in travel time from a sender to pairs of receivers, respectively. The two types of measurements can also be thought of as absolute and relative time distances, or delays [42]. An example of automatic calibration using TDOA measurements is shown in Figure 1.2.

Once we have collected data – for example TOA measurements or images – and estimated a map from this using SfM or automatic calibration, the map can be used for *localisation*. For vision, localisation would mean to find the position of a camera, given an image of the already mapped environment. This can for example be useful for an agent that returns to a scene that has previously been visited. It can also be useful for path planning. In the case of sound signals localisation often refers to determining where the sound source is located, given the signal and the position of the microphones [12, 18, 19, 26]. A group of sensors that are placed at different positions can also be referred to as a *sensor network*.

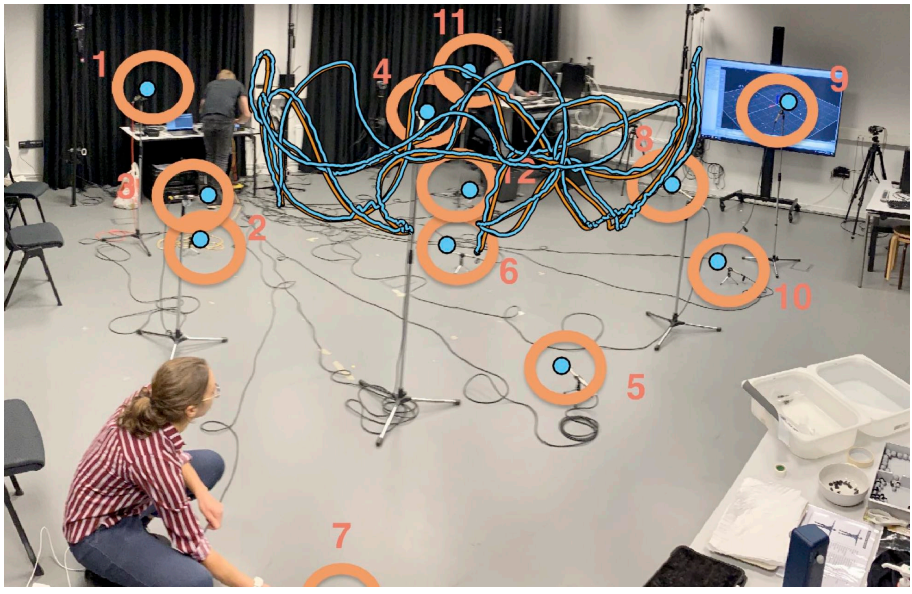


Figure 1.2: A setup consisting of twelve microphones that were used to collect TDOA measurements from a moving sound source. Both the receiver and sender positions were calculated using automatic calibration. The reconstructed sound source path is shown as a blue curve and the ground truth path in orange. The orange circles indicate where the microphones were placed and the blue dots show the estimated receiver positions.

Except for mapping and localisation, other usages of sensor networks are, for example, sound quality improvement using beam-forming [2] and speaker diarisation – used for example to determine who spoke when [3].

Localisation in a known environment will be easier the more accurate the map is. Because of this, it can be a good thing if information can be added to an already existing map, or if several different map representations of the same scene can be *merged* or *fused* into a single, more accurate, map. One example for when this could be useful is the following: Imagine that every car that is driving through a city can build its own, local map of the city using its collected image data and SfM/SLAM techniques. The result would be a large amount of representations of the same city map. If we have a fast and accurate way to merge individual map representations into one global map, each car could contribute to that global map. The city map would thus improve with each car passing. This also means that changes in the infrastructure could be added, without the map being re-made. Figure 1.3 shows three local maps of a room and a merge of these into a global map which covers more of the room and is more accurate. The merge actually consists of several parts, such as identifying which regions that are the same in the local maps; to match or align the different maps; and to then do the actual merge of the parts that are the same.

The problem of map merging has a strong connection to loop closure, where any drift that has arisen in the SfM process has to be adjusted for when the camera returns to a position

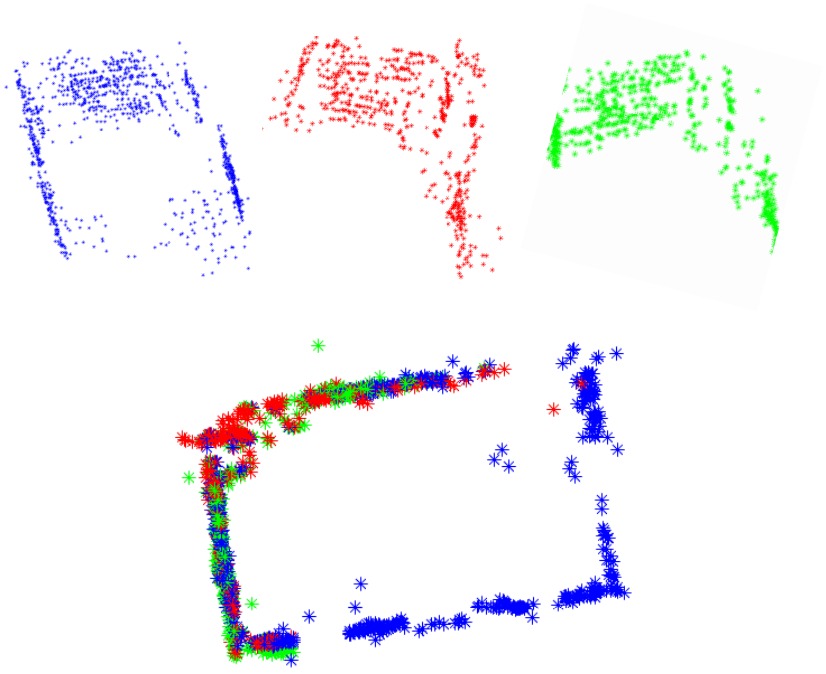


Figure 1.3: The three images on top – the blue, red and green point clouds – show three different 3D reconstructions or local maps of a room obtained using SfM. In the plot below three such local maps have been merged into one, global map of the room.

that has already been visited [97]. This corresponds to a merge of the start and the end of the map. The map fusion issue has also been addressed within the field of collaborative SLAM, where several cameras are simultaneously used for SLAM. There are for example applications with several drones where the merge is based on a few keyframes [78]. This is fast, but is highly dependent on the choice of keyframes. In some other cases the problem has been simplified by common initialisation [111], or by mounting the cameras such that their relative position is fixed [66].

In this thesis we investigate several ways of achieving accurate maps. The included papers can be divided in three parts: Paper I which has a focus on finding very accurate TDOA measurements; Papers II–V that focus on the robust self-calibration problem for one-dimensional signals; and Papers VI–VIII that investigate how matching and merging of maps can be done in a fast but accurate way.

Before the papers are presented, the introductory part of the thesis will be organised as follows: In Chapter 2 some methodology will be introduced and in Chapter 3 we go through sensor modelling. In Chapter 4 we show how sensor data can be used for mapping and loc-

alisation, including SfM. Chapter 5 treats the problem of map merging and in Chapter 6 we present a summary of the results from the included papers and discuss how this work can be developed in the future.

Chapter 2

Methodology

Signals are fundamental for the studies in this thesis. A signal can be described as the entity that carries some sort of information from one point to another [36, p. 1]. It can also be viewed as any physical quantity that varies with time, space or one or several other independent variables [72, p. 2]. Parts of this thesis regard the study and analysis of audio, that is, information transmitted as an acoustic wave through space. This is an example of a *one-dimensional signal*. Other examples of such are radio signals, for example ultra wide-band (UWB), which is used in Paper II. UWB is a radio technology that transmits information short distances using a wide frequency band and low energy [84]. There are also signals of higher dimensions than these, such as *two-dimensional images*. An image consists of one or several channels, where each pixel in a channel is a number representing an intensity. Examples of signals with three or four dimensions are, respectively, voxel represented images used in for example medicine and such voxel images with an additional dimension that corresponds to change over time. This thesis will, however, only treat one- and two-dimensional signals.

Even if we study several different signal realisations, many principles are the same for them all and both methods and applications can be adjusted slightly such that another signal type can be used. Therefore, parts of this thesis will discuss signals in general, while parts will be more focused on either audio or images.

2.1 One-Dimensional Signal Processing

Most of the signals that we model and analyse are in reality analogue, but for analysis they need to be converted into a digital format. For the one-dimensional case this is done through sampling. Assume that we have an analogue signal $\mathbf{x}_a(t)$, $t \in \mathbb{R}$, and denote the

discretisation operator by $D : \mathbb{B} \to \ell$. Here, $\mathbb{B}$ are functions $f \in C(\mathbb{R}, \mathbb{R})$ that are square integrable with vanishing Fourier transform outside $[-\pi, \pi]$, while ℓ denotes the set of discrete, square summable functions from $\mathbb{Z}$ to $\mathbb{R}$. An analogue signal can thus be sampled by

$$\mathbf{x}(n) = D(\mathbf{x}_a)(n) = \mathbf{x}_a(nT), \tag{2.1}$$

where T is the period, that is, a sample is taken every T seconds, and $x(n)$ with $n \in \mathbb{Z}$ is the digital signal corresponding to $\mathbf{x}_a(t)$. The sampling period also defines the sampling rate or frequency, $F_s = 1/T$.

As long as the sampling frequency is at least twice as big at the highest frequency $F_{\max}$ in the signal, $F_s > 2 \cdot F_{\max}$, the *sampling theorem* states that the analogue $\mathbf{x}_a(t)$ can be recovered exactly from its digital representation $\mathbf{x}(n)$ [44, 67, 72, 81, 96]. The recovered analogue signal $\hat{\mathbf{x}}_a$ can be obtained by interpolation of $\mathbf{x}$ with a kernel $\mathbf{g}$. If we denote the interpolation operator $I_{\mathbf{g}} : \ell \to \mathbb{B}$, we have

$$\hat{\mathbf{x}}_a(t) = I_{\mathbf{g}}(\mathbf{x})(t) = \sum_{i=-\infty}^{\infty} \mathbf{g}(t-i)\mathbf{x}(i). \tag{2.2}$$

With $\mathbf{g}$ given by the normalised sinc operator

$$\operatorname{sinc}(k) = \begin{cases} \frac{\sin(\pi k)}{\pi k}, & \text{if } x \neq 0, \\ 1, & \text{if } x = 0, \end{cases} \tag{2.3}$$

we have that

$$I_{\operatorname{sinc}}(D(\mathbf{x}_a)) = \mathbf{x}_a. \tag{2.4}$$

Hence, $\hat{\mathbf{x}}_a(t) = \mathbf{x}_a(t)$ if $\hat{\mathbf{x}}$ is obtained using $\mathbf{g} = \operatorname{sinc}$. This is referred to as ideal interpolation.

In many cases a sampled signal does, however, also contain noise. Then, the connection could rather be described as

$$\tilde{\mathbf{x}}(n) = \mathbf{x}(n) + \mathbf{e}(n) = D(\mathbf{x}_a)(n) + \mathbf{e}(n), \tag{2.5}$$

where $\mathbf{e}(n)$ describes the noise and $\tilde{\mathbf{x}}(n)$ is the digital signal that is used for analysis. The noise can have several origins, such as other signals that were not supposed to be captured or added noise that has arisen in the sampling process. For an audio signal this could be someone speaking in the background, or disturbances in the microphone.

Noise coming from disturbances often has a high frequency. To remove some of that noise one can apply a filter to the sampled signal, in order to smooth out the high-frequency components. Usually the signal itself contains lower frequencies, so most of the information in the signal can be kept through the filtering. Furthermore, patterns on a coarser scale are

easier captured after smoothing [57]. In this thesis, smoothing will refer to interpolation with the Gaussian kernel

$$\mathbf{G}_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{x^2/(2\sigma^2)}. \tag{2.6}$$

The value σ is the standard deviation of the Gaussian function and determines the width of the kernel. The higher the value of σ is, the more smoothing.

Furthermore, there is another advantage of smoothing with a Gaussian function when ideal interpolation is employed. It turns out that interpolation with a Gaussian kernel followed by interpolation with the sinc kernel can be approximated by only Gaussian interpolation [4]. What we get is

$$\hat{\mathbf{x}}_a(t) = (\mathbf{G}_\sigma * I_{\text{sinc}}(\mathbf{x}))(t) = I_{\mathbf{G}_\sigma * \text{sinc}}(\mathbf{x})(t) \approx I_{\mathbf{G}_\sigma}(\mathbf{x})(t). \tag{2.7}$$

However, this only holds when the standard deviation σ is *large enough*. How large it has to be is studied in Paper I.

2.2 Two-Dimensional Signal Processing

In many ways, what was presented in the previous section concerning one dimensional signals is also relevant for two dimensional signals. Therefore, the presentation in this section will be slightly briefer. As in the case of a one-dimensional signal, a two-dimensional signal has to be discretised in order for it to be processable. An example of a discretised, two-dimensional signal is an image $\mathbf{X}$. Each pixel of the image represents the intensity from a certain area in the real world. The corresponding continuous signal is denoted $\mathbf{X}_a$. A (continuous) greyscale image can be seen as a function $\mathbf{X}_a : \Omega \to \mathbb{R}_+$, where $\Omega = \{(x,y) | a \leq x \leq b, c \leq y \leq d\} \subseteq \mathbb{R}^2$ and $\mathbb{R}_+$ are the non-negative real numbers. Sometimes, mainly to simplify the theory, the limits of the domain are ignored, and we let $\Omega = \mathbb{R}^2$, similar to what we did in the previous section. Furthermore, in order to get the discrete image $\mathbf{X}$ the variables x and y are discretised in a chosen number of rows m and columns n and $m \cdot n$ gives the total number of pixels – sometimes denoted $m \times n$. To down- or up-sample the image, the values of m and n can be decreased or increased, respectively. For a coloured image there are several channels; an RGB image, for example, has a third dimension to represent the intensity values belonging to the red, green and blue primary colours separately. Hence, the discrete RGB image is a function $\mathbf{X}(i,j,k)$. For image processing, each of these channels can be treated separately, and for simplicity this section will only treat greyscale images, that is functions $\mathbf{X}(i,j)$.

A discrete image $\mathbf{X}$ can be interpolated in order to recover the continuous signal. Hence, if the image $\mathbf{X}(i,j)$ is a discretisation of $\mathbf{X}_a$, then

$$\mathbf{X}(i,j) = D(\mathbf{X}_a)(i,j) = \mathbf{X}_a(iT, jT), \tag{2.8}$$

with T being sampling width. The converse, interpolation, is defined as

$$\hat{\mathbf{X}}_a(x,y) = I_g(\mathbf{X})(x,y) = \sum_{k=-\infty}^{\infty} \sum_{l=-\infty}^{\infty} \mathbf{g}(x-k, y-l)\mathbf{X}(k,l), \tag{2.9}$$

where $\mathbf{g}$ again is the interpolation kernel – here in two dimensions. For sinc in two dimensions we get

$$\mathbf{g}(x,y) = \text{sinc}(x)\text{sinc}(y), \tag{2.10}$$

and as before, this ideal interpolation gives that for square integrable, bounded functions with Fourier transform that is zero outside $[-\pi,\pi] \times [-\pi,\pi]$ the original signal is restored [4],

$$I_{\text{sinc}}(D(\mathbf{X}_a)) = \mathbf{X}_a. \tag{2.11}$$

If we decide to include measurement errors – for example coming from the camera – in the model, we could describe the digital image as

$$\tilde{\mathbf{X}}(i,j) = \mathbf{X}(i,j) + \mathbf{E}(i,j) = D(\mathbf{X}_a)(i,j) + \mathbf{E}(i,j), \tag{2.12}$$

where $\mathbf{E}(i,j)$ is the noise in the different pixels. As in the previous section, an image can be smoothed through convolution with a Gaussian kernel

$$\mathbf{G}_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \mathrm{e}^{-(x^2+y^2)/(2\sigma^2)}, \tag{2.13}$$

with σ being the standard deviation in both directions. This action can decrease the impact of measurement noise. There are also several other kernels that can have this effect, depending on the character of the noise.

2.3 Statistical Measures

In this section we will begin by mentioning some statistical measures for one-dimensional signals. For a more thorough explanation, see [58]. We denote one such signal $\mathbf{x}(n)$ and let this be a stochastic variable signal. Given this, the expected value $\mathbf{E}[\cdot]$ is defined as

$$\mathbf{E}[\mathbf{x}(n)] = \sum a p_{\mathbf{x}(n)}(a), \tag{2.14}$$

where $p_{\mathbf{x}(n)}(a) = \mathrm{P}(\mathbf{x}(n) = a)$ is the probability function. Using the expected value, the variance of a signal is

$$\mathbf{V}[\mathbf{x}(n)] = \mathbf{E}[(\mathbf{x}(n) - \mathbf{E}[\mathbf{x}(n)])^2] = \mathbf{E}[\mathbf{x}(n)^2] - \mathbf{E}[\mathbf{x}(n)]^2, \tag{2.15}$$

and the standard deviation is the square root of the variance,

$$\sigma_{\mathbf{x}}(n) = \sqrt{\mathbf{V}[\mathbf{x}(n)]}. \tag{2.16}$$

For the connection between two signals $\mathbf{x}(n)$ and $\mathbf{y}(n)$, the covariance is defined as

$$\begin{aligned}\mathbf{C}[\mathbf{x}(n), \mathbf{y}(n)] =& \mathbf{E}[(\mathbf{x}(n) - \mathbf{E}[\mathbf{x}(n)])(\mathbf{y}(n) - \mathbf{E}[\mathbf{y}(n)])] \\ =& \mathbf{E}[\mathbf{x}(n)\mathbf{y}(n)] - \mathbf{E}[\mathbf{x}(n)]\mathbf{E}[\mathbf{y}(n)].\end{aligned} \tag{2.17}$$

The covariance is such that $\mathbf{C}[\mathbf{x}(n), \mathbf{x}(n)] = \mathbf{V}[\mathbf{x}(n)]$. For a single process, the covariance function is

$$r(n, m) = \mathbf{C}[\mathbf{x}(n), \mathbf{x}(m)]. \tag{2.18}$$

Furthermore, the cross-correlation between two signals is [42]

$$(\mathbf{x} \star \mathbf{y})(n) = \mathbf{E}[\mathbf{x}(a)\mathbf{y}(a + n)]. \tag{2.19}$$

The cross-correlation is also a valuable measure for deterministic signals, for which it is defined as

$$(\mathbf{x} \star \mathbf{y})(n) = \sum_a \mathbf{x}(a)\mathbf{y}(a + n). \tag{2.20}$$

This is a common way to measure at which delay the two signals are most common, see Section 3.1.3.

A stochastic process that has constant mean and a covariance function that only depends on the delay – that is, $r(n, m) = \hat{r}(n - m)$ – is called a weakly stationary or wide-sense stationary (WSS) process [58]. This means that for example the mean in Equation (2.14) and the standard deviation in Equation (2.16) do not depend on n. This is a common assumption. In this thesis, noise will be assumed to be WSS and have expected value zero.

Using the constant mean of a function $\mathbf{E}[\mathbf{x}]$ and the noise $\mathbf{E}[\mathbf{e}]$, the noise level in a signal can be measured through the *signal-to-noise ratio* (SNR). The SNR can be computed in different ways, depending on what information that is known about the signal. Some definitions are [58]

$$\text{SNR} = \frac{P_{\text{signal}}}{P_{\text{noise}}} = \frac{\mathbf{E}[\mathbf{x}^2]}{\mathbf{E}[\mathbf{e}^2]} = \frac{\mathbf{E}[\mathbf{x}^2]}{\sigma_{\mathbf{e}}^2}, \tag{2.21}$$

where P_{signal} and P_{noise} is the power of the signal and the noise, respectively, and $\sigma_{\mathbf{e}}$ is the standard deviation of the noise. Since we assume that the noise is WSS, $\mathbf{E}[\mathbf{e}^2] = \sigma_{\mathbf{e}}^2$. Also, in most cases the signal is deterministic, hence $\mathbf{E}[\mathbf{x}^2] = \mathbf{x}^2$, and the fraction simplifies to $\mathbf{x}^2/\sigma_{\mathbf{e}}^2$. If the signal is stochastic and with mean zeros as well, we get that

$$\text{SNR} = \frac{\sigma_{\mathbf{x}}^2}{\sigma_{\mathbf{e}}^2}, \tag{2.22}$$

with $\sigma_{\mathbf{x}}$ being the standard deviation of the signal.

The statistical measures above are defined analogously for two dimensional signals. Firstly, each such signal can be viewed as a one dimensional signal by column stacking the pixel measurements

$$\mathbf{x}(a+(b-1)m) = \mathbf{X}(a,b), \qquad a = 1,\dots,m,\ b = 1,\dots,n. \tag{2.23}$$

This column stacking can also be denoted $\mathbf{x} = \mathbf{X}(:)$. Through this, the mean and variance of an image can be computed as described in Equations (2.14) and (2.15). Concerning the cross-correlation between two images $\mathbf{X}$ and $\mathbf{Y}$, this is defined as

$$(\mathbf{X} \star \mathbf{Y})(i,j) = \sum_a \sum_b \mathbf{X}(i+a, j+b)\mathbf{Y}(a,b). \tag{2.24}$$

2.3.1 Matrix Decompositions

In this section some useful matrix decompositions, such as eigenvalue decomposition and QR factorisation, will be mentioned. We will also briefly mention PCA analysis. For more information on these matters, see [90], from which the rest of this section has been inspired.

Singular Value Decomposition

Any real-valued $m \times n$ matrix $\mathbf{X}$ can be divided into a *singular value decomposition* (SVD)

$$\begin{aligned}\mathbf{X} &= \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \\ &= \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_p\end{bmatrix} \begin{bmatrix}\sigma_1 & & & \\ & \sigma_2 & & \\ & & \ddots & \\ & & & \sigma_p\end{bmatrix} \begin{bmatrix}\mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T\end{bmatrix} = \sum_{i=1}^{p} \sigma_i \mathbf{u}_i \mathbf{v}_i^T,\end{aligned} \tag{2.25}$$

where $p = \min(m,n)$, $\mathbf{U}$ is an orthonormal $m \times p$ matrix, $\mathbf{V}$ is an orthonormal $n \times p$ matrix and $\mathbf{\Sigma}$ is a diagonal $p \times p$ matrix. The *singular values* σ_i are all non-negative and the number of positive singular values is the same as the rank r of $\mathbf{X}$. If $r < p$, the last $p - r$ columns of $\mathbf{U}$ and $\mathbf{V}$ can be neglected. As an example, this means that for a 5×6 matrix A of rank 3, we can express it as $\mathbf{U}\mathbf{\Sigma}\mathbf{V}$, where U is 5×3, $\mathbf{V}$ is 6×3 and $\mathbf{\Sigma}$ is 3×3.

Furthermore, a matrix $\mathbf{X}$ can be compressed and approximated with a lower rank matrix

$\tilde{\mathbf{X}}$ by decreasing the upper limit of the sum in Equation (2.25), that is, by letting

$$\tilde{\mathbf{X}} = \begin{bmatrix}\mathbf{u}_1 & \mathbf{u}_2 & \dots & \mathbf{u}_p\end{bmatrix} \begin{bmatrix} \sigma_1 & & & & & \\ & \ddots & & & & \\ & & \sigma_k & & & \\ & & & 0 & & \\ & & & & \ddots & \\ & & & & & 0 \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \mathbf{v}_2^T \\ \vdots \\ \mathbf{v}_p^T \end{bmatrix} = \sum_{i=1}^{k} \sigma_i \mathbf{u}_i \mathbf{v}_i^T, \quad (2.26)$$

we get a k rank approximation. This will give the best possible approximation to the original matrix, in a least squares sense [90]. Note that due to the zeroes, we might as well neglect the last columns of $\mathbf{U}$ and $\mathbf{V}$ and write this as

$$\tilde{\mathbf{X}} = \begin{bmatrix}\mathbf{u}_1 & \dots & \mathbf{u}_k\end{bmatrix} \begin{bmatrix} \sigma_1 & & \\ & \ddots & \\ & & \sigma_k \end{bmatrix} \begin{bmatrix} \mathbf{v}_1^T \\ \vdots \\ \mathbf{v}_k^T \end{bmatrix}. \quad (2.27)$$

Eigenvalue Decomposition

Given a square $n \times n$ matrix $\mathbf{X}$, it has the *eigenvectors* $\mathbf{p}_i$ and *eigenvalues* λ_i if these satisfy

$$\lambda_i \mathbf{p}_i = \mathbf{X}\mathbf{p}_i. \quad (2.28)$$

If $\mathbf{X}$ is of full rank, there will be n linearly independent eigenvectors. If so is the case it can be decomposed as

$$\begin{aligned} \mathbf{X} &= \mathbf{P}\mathbf{\Lambda}\mathbf{P}^T \\ &= \begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \dots & \mathbf{p}_n\end{bmatrix} \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_n \end{bmatrix} \begin{bmatrix} \mathbf{p}_1^T \\ \mathbf{p}_2^T \\ \vdots \\ \mathbf{p}_n^T \end{bmatrix} = \sum_{i=1}^{n} \lambda_i \mathbf{p}_i \mathbf{p}_i^T. \end{aligned} \quad (2.29)$$

If $\mathbf{X}$ is symmetric the eigenvalues and eigenvectors only contain real numbers, while non-symmetric matrices can give complex entries [90].

When doing eigenvalue decomposition of a covariance matrix, this is referred to as *principal component analysis* (PCA). The covariance $\mathbf{C}$ of a set of points $\mathbf{x}_i$ (potentially in several dimensions) is

$$\mathbf{C} = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i - \bar{\mathbf{x}})(\mathbf{x}_i - \bar{\mathbf{x}})^T, \quad (2.30)$$

where $\bar{\mathbf{x}}$ denotes the mean of the points (this is similar to Equation (2.17)). This matrix $\mathbf{C}$ is positive semi-definite and all its eigenvalues are non-negative. If these eigenvalues are ordered in decreasing size, the eigenvector corresponding to the largest eigenvalue will represent the direction in which the covariance is largest, that is, where there is most variation for the points $\mathbf{x}_i$. The eigenvectors of a covariance matrix can also be called the *modes of variation*.

QR Decomposition

The *QR decomposition* or *QR factorisation* of a matrix $\mathbf{X}$ is a factorisation

$$\mathbf{X} = \mathbf{QR}, \tag{2.31}$$

where $\mathbf{Q}$ is an orthonormal matrix (satisfying $\mathbf{QQ}^T = \mathbf{I}$) and $\mathbf{R}$ is upper triangular. If we use such a factorisation, the square of a matrix can be expressed as

$$\mathbf{X}^T\mathbf{X} = \mathbf{R}^T\mathbf{Q}^T\mathbf{QR} = \mathbf{R}^T\mathbf{R}. \tag{2.32}$$

This has been used in several of the papers of this thesis. Analogously, we can define the *RQ factorisation*, where the matrix order is reversed

$$\mathbf{X} = \mathbf{RQ}, \tag{2.33}$$

that is, the upper triangular matrix $\mathbf{R}$ is multiplied with the orthogonal matrix $\mathbf{Q}$ from the right. RQ factorisation can be used to divide a camera matrix into a rotation matrix and an upper triangular calibration matrix, see Section 4.3.1 [35].

2.4 Minimal Solvers

Many of the papers in this thesis – and tasks in computer vision and automatic calibration in general – concern solving some problem which can be defined by one or several equations. There are different ways of finding an optimal solution to such a problem. One way could be to first find a reasonable – but not necessarily optimal – initial solution for the problem, and then refine that solution. Another way could be to solve a sub-problem which is small enough for us to find an exact, optimal solution to, and then extend that solution to the rest of the problem. In this section, the concept of *minimal solvers* will be explained.

Given a system of polynomial equations, a *minimal problem* is a case where the effective number of degrees of freedom is the same as the number of variables to solve for. A minimal solver is a solver that uses the minimal amount of data, that is, that solves a minimal problem using just enough data. In most cases we also require the number of solutions to be finite.

This is for practical reasons; in computer vision we often want to go through all solutions to find the one that suits the problem best, and this is not possible if there are no or infinitely many solutions. Hence, we search for minimal problems with a finite number of solutions.

One example of a minimal problem is to fit a line to two points in 2D or 3D and another is to fit a plane to three points in 3D. For the latter, if only two points are given, the plane is not uniquely determined and given four arbitrary points it is in general not possible to find a plane that goes through them all. Given the three points (x_1, x_2, x_3), (y_1, y_2, y_3) and (z_1, z_2, z_3) the system of equations to solve would be

$$\begin{cases} x_1 + ax_2 + bx_3 + c = 0, \\ y_1 + ay_2 + by_3 + c = 0, \\ z_1 + az_2 + bz_3 + c = 0. \end{cases} \tag{2.34}$$

There are in total three degrees of freedom (one from each equation) and there are three variables to solve for (a, b and c).

Solutions to minimal problems can often be found using methods from algebraic geometry. In this section a brief introduction to the field will be given. For more details, see [20, 21, 51], from which this section has been inspired.

2.4.1 Basics of Algebraic Geometry

In this section we will use a number of variables $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$. For these, a *monomial* is a finite product

$$\mathbf{x}^{\boldsymbol{\alpha}} = x_1^{\alpha_1} \cdot x_2^{\alpha_2} \cdot \ldots \cdot x_n^{\alpha_n}. \tag{2.35}$$

It is said to have degree $|\boldsymbol{\alpha}| = \alpha_1 + \ldots + \alpha_n$. Let $\mathbb{K}$ denote a field, for example $\mathbb{R}$ or $\mathbb{C}$. A *polynomial* f in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is a finite linear combination of monomials,

$$f(\mathbf{x}) = \sum_{\boldsymbol{\alpha}} a_{\boldsymbol{\alpha}} \mathbf{x}^{\boldsymbol{\alpha}}, \quad a_{\boldsymbol{\alpha}} \in \mathbb{K}. \tag{2.36}$$

For $f(\mathbf{x})$ to be a polynomial we require that the sum is finite, that is, that there are only finitely many $\boldsymbol{\alpha}$ such that $a_{\boldsymbol{\alpha}} \neq 0$. Indirectly, this means that we could change the sum such that we only sum over the terms for which $a_{\boldsymbol{\alpha}}$ is non-zero. Using a set of such polynomials, a system of equations can be described as

$$\begin{cases} f_1(\mathbf{x}) = 0, \\ f_2(\mathbf{x}) = 0, \\ \quad \vdots \\ f_m(\mathbf{x}) = 0. \end{cases} \tag{2.37}$$

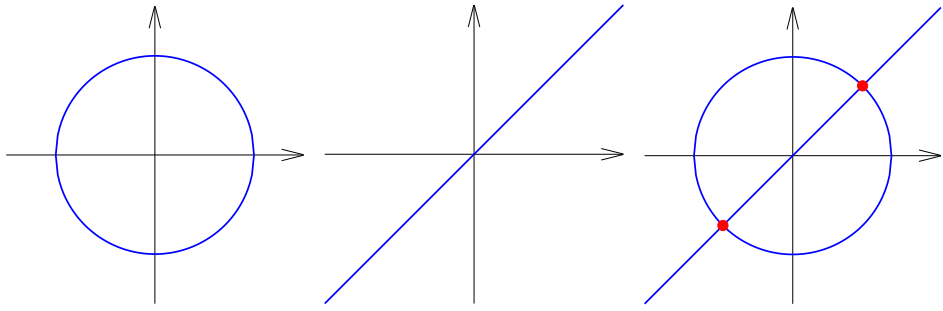


Figure 2.1: For $f_1 = x^2 + y^2 - 1$ and $f_2 = x - y$ the leftmost plot shows the affine variety $V(f_1)$ in blue and the plot in the middle show the variety $V(f_2)$, also in blue. The red dots in the rightmost plot shows $V(f_1, f_2)$.

The set of all polynomials in $\mathbf{x}$ with coefficients in $\mathbb{K}$ is denoted $\mathbb{K}[\mathbf{x}]$.

It is common to write a polynomial with the largest monomial first followed by monomials in decreasing order. For a single variable this is trivial – the largest monomial is that of highest degree, and this monomial is unique – but for polynomials of several variables there can be several monomials of the same degree and it is not obvious which of these is the largest. There are different ways of ordering such monomials, that is, different rules to decide which monomial is the largest. Examples of monomial orderings are lexiographical ordering (lex), graded lexiographical ordering (grlex) and graded reverse lexiographical ordering (grevlex), see [20]. Once an ordering has been chosen, we can define the *leading term* of a polynomial $\mathrm{LT}(f)$ to be the term (including the coefficient) containing the largest monomial with respect to the ordering.

Furthermore, the *affine variety* $V(f_1, \ldots, f_m) \subset \mathbb{K}^n$ is the set of all points $\mathbf{x}$ such that $f_i(\mathbf{x}) = 0$ for all $i = 1, \ldots, m$,

$$V(f_1, \ldots, f_m) = \{(a_1, \ldots, a_n) \in \mathbb{K}^n;\, f_i(a_1, \ldots, a_n) = 0 \;\forall\; i = 1, \ldots, m\}. \tag{2.38}$$

Hence, this affine variety represents all solutions to the system of Equations (2.37). With $\mathbb{K} = \mathbb{R}, f_1 = x^2 + y^2 - 1$ and $f_2 = x - y$, $V(f_1)$ represents the unit circle, and $V(f_1, f_2)$ consists of the two points $(-1/\sqrt{2}, -1/\sqrt{2})$ and $(1/\sqrt{2}, 1/\sqrt{2})$. These are illustrated in Figure 2.1.

Moreover, the *ideal* generated by a set of polynomials $(f_1, \ldots, f_m)$ is defined as

$$\langle f_1, \ldots, f_m \rangle = \left\{ \sum_{i=1}^{m} h_i f_i;\; h_1, \ldots, h_m \in \mathbb{K}[\mathbf{x}] \right\}, \tag{2.39}$$

and the polynomials generating the ideal are called the basis of the ideal. An ideal is such

that

$$\begin{cases} 0 \in I, \\ f_1, f_2 \in I \Rightarrow f_1 - f_2 \in I, \\ f \in I,\ h \in \mathbb{K}[\mathbf{x}] \Rightarrow hf \in I. \end{cases} \tag{2.40}$$

The set in Equation 2.39 has these properties, but any set of polynomials $\{f_i\}$ that fulfills these the requirements is an ideal, so the ideal does not necessarily have to be created as given in Equation 2.39. However, it can be shown that every ideal can be generated by some (not unique) finite basis. Furthermore, we can form the ideal of an affine variety, $I(V)$, which will be the set of all polynomials f for which $f(V) = 0$ (that is, $f(v) = 0$ for all $v \in V$). The *leading term* of an ideal I is the collection of all the leading terms of elements of I, that is

$$\mathrm{LT}(I) = \{a_\alpha \mathbf{x}^\alpha;\ \text{there exist}\ f \in I\ \text{with}\ \mathrm{LT}(f) = a_\alpha \mathbf{x}^\alpha\}. \tag{2.41}$$

Given a monomial order, a subset $G = \{g_1, g_2, \ldots, g_m\}$ of an ideal I is a *Groebner basis* if

$$\langle \mathrm{LT}(g_1), \ldots, \mathrm{LT}(g_m) \rangle = \langle \mathrm{LT}(I) \rangle, \tag{2.42}$$

that is, if the leading terms of the Groebner basis generate the same ideal as all the leading terms of any polynomial generating I. One can also think of this as G being a Groebner basis if for any $f \in I$, $\mathrm{LT}(f)$ is divisible by some $\mathrm{LT}(g_i)$. This means that a Groebner basis for an ideal I is also a basis for the ideal. Furthermore, any ideal has a Groebner basis. A Groebner basis for an ideal I can be found for example using the Buchberger's Algorithm or improvements thereof [20].

Given any ordered m-tuple $F = (f_1, \ldots, f_m))$ of polynomials in $\mathbb{K}[\mathbf{x}]^m$ and a chosen monomial order, the *division algorithm* states that every $f \in \mathbb{K}[\mathbf{x}]$ can be written

$$f = a_1 f_1 + \ldots + a_m f_m + r. \tag{2.43}$$

Both $a_i, r \in \mathbb{K}[\mathbf{x}]$ and the *remainder* r is either zero or such that no monomials in r are divisible by any of $\mathrm{LT}(f_1), \ldots, \mathrm{LT}(f_m)$. One issue with the division algorithm is that for a general polynomial f and set of polynomials F, the remainder will be dependent on the order of the polynomials in F. However, if the polynomial f is divided by a set of polynomials $G = \{g_1, \ldots, g_m\}$ which is a Groebner basis, the remainder will be unique and independent of the order of g_i. This also gives that the polynomial $f \in I$ if and only if $r = 0$ after division by a Groebner basis G. The remainder of f after division by G is also denoted $\overline{f}^G$. This is also called the *normal form* of $f \in \mathbb{K}[\mathbf{x}]$ with respect to G. For more details on the division algorithm, see [20, 51].

Furthermore, we define the *quotient ring* $\mathbb{K}[\mathbf{x}]/I$ associated with the ideal I as the set of equivalence classes such that

$$a \sim b \Leftrightarrow a \equiv b \mod I \Leftrightarrow a - b \in I. \tag{2.44}$$

The equivalent class that a certain polynomial a belongs to is denoted $[a]$ and hence $[a] = \{b \in \mathbb{K}[\mathbf{x}];\ a - b \in I\}$. This means that every polynomial that is in the ideal is zero in the equivalent class, and for polynomials that are not in the ideal, we only save the remainder after division with the basis of the ideal. Considering an ideal generated by a uni-variate polynomial of degree n, $I = \langle f(x) \rangle$, with $f = \sum_{i=1}^{n} a_i x^i$, the quotient ring $\mathbb{K}[x]/I$ will be spanned by the monomials of degree lower than n, that is, $\{x^{n-1}, \ldots, x, 1\}$. Monomials x^i such that $\{x^i;\ x^i \notin \langle \mathrm{LT}(I) \rangle\}$ are called the *standard monomials*. The same applies to polynomials of several variables, that the monimials that cannot be reduced further are called standard monomials. This would be the monomials $\{\mathbf{x}^{\boldsymbol{\alpha}};\ \mathbf{x}^{\boldsymbol{\alpha}} \notin \langle \mathrm{LT}(I) \rangle\}$.

2.4.2 Solving Systems of Equations

There are different ways of solving systems of polynomial equations using algebraic geometry. Many of them seeks to transform the original problem into an eigenvalue problem, see Section 2.3.1. In this section we will give a brief overview of resultant based methods and the action matrix method. For a more thorough explanation for these and other methods, see [21, 51].

Resultant Based Method

Given two uni-variate polynomials

$$\begin{aligned} f(x) &= a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0, \quad a_n \neq 0 \\ g(x) &= b_m x^m + b_{m-1} x^{m-1} + \ldots + b_1 x + b_0, \quad b_m \neq 0, \end{aligned} \tag{2.45}$$

the *resultant* is the determinant of a $(m+n) \times (m+n)$ matrix, defined as

$$\mathrm{Res}(f,g) = \det \begin{bmatrix} a_n & \ldots & \ldots & a_0 & & & \\ & a_n & \ldots & \ldots & a_0 & & \\ & & \ddots & & & \ddots & \\ & & & a_n & \ldots & \ldots & a_0 \\ b_m & \ldots & b_0 & & & & \\ & b_m & \ldots & b_0 & & & \\ & & \ddots & & \ddots & & \\ & & & b_m & \ldots & b_0 & \end{bmatrix}. \tag{2.46}$$

The first m rows contain the coefficients of the f polynomial and the last n rows the coefficients of g. The resultant is such that if f and g have a common root, $\mathrm{Res}(f,g) = 0$. Hence, the resultant can be used to find such roots, for a uni-variate pair of equations. Furthermore, if there are two unknowns, one of them can be considered as a coefficient and be

included in the resultant to give a constraint for a common solution. This is referred to as the *hidden variable trick*. The resultant method is a good theoretical instrument, but as the systems of equations become large, with several polynomials and unknowns, solving the equation $\text{Res}(f, g) = 0$ becomes hard and therefore it is not applicable to all problems.

Action Matrix Methods

Again, consider a polynomial in $\mathbb{K}[\mathbf{x}]$ in one variable, for which the coefficient of the highest term is one,

$$f(x) = x_n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0. \tag{2.47}$$

The *companion matrix* connected to $f(x)$ is

$$C = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \ldots & -a_0 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}. \tag{2.48}$$

and the characteristic polynomial of this matrix is equal to the polynomial in Equation 2.47. Hence, the roots of the polynomial are given by the eigenvalues of the companion matrix. This can be used to solve the equation $f(x) = 0$.

Furthermore, consider the map $T_\alpha : \mathbb{K}[\mathbf{x}]/I \to \mathbb{K}[\mathbf{x}]/I$ which multiplies a polynomial $p(\mathbf{x}) \in \mathbb{K}[\mathbf{x}]$ with a fixed monomial (or polynomial) α, $T_\alpha([p(\mathbf{x})]) = [\alpha p(\mathbf{x})]$. In the case above, we have $T_\alpha : \mathbb{K}[x]/\langle f(x)\rangle \to \mathbb{K}[x]/\langle f(x)\rangle$. If we let $\alpha = x$ and also define the basis for $\mathbb{K}[x]/\langle f(x)\rangle$ as $\mathbf{b} = \{b_1, \ldots, b_n\} = \{x^{n-1}, \ldots, x, 1\}$ we see that we can express $T_\alpha(\mathbf{b}) = T_x(\mathbf{b})$ using the companion matrix

$$T_x(\mathbf{b}) = x \begin{bmatrix} x^{n-1} \\ \vdots \\ x \\ 1 \end{bmatrix} = \begin{bmatrix} x^n \\ \vdots \\ x^2 \\ x \end{bmatrix} = \begin{bmatrix} -a_{n-1} & -a_{n-2} & \ldots & -a_0 \\ 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix} \begin{bmatrix} x^{n-1} \\ \vdots \\ x \\ 1 \end{bmatrix}, \tag{2.49}$$

since $[x^n] = [-a_{n-1}x^{n-1} - \ldots - a_1 x - a_0]$ in $\mathbb{K}[x]/\langle f\rangle$. As can be seen, the value for x can be found as an eigenvalue to the companion matrix, as well as within the (correctly normalised) eigenvector $\mathbf{b}$.

This can be generalised to polynomials of more variables. If we have a system of equations as in Equation (2.37) the solution is given by the shared roots of $f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})$. Again, let I

be the ideal generated by the set of polynomials, $I = \langle f_1, \ldots, f_m \rangle$, and let $\mathbf{b} = \{b_1, \ldots, b_n\}$ be a basis for $\mathbb{K}[\mathbf{x}]/I$. Similar to above, we can express the operator T_α (now with a general monomial/polynomial α) using an unknown $n \times n$ matrix $\mathbf{M}$

$$T_\alpha(\mathbf{b}) = \alpha \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & \ldots & m_{1n} \\ m_{21} & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ m_{n1} & \ldots & \ldots & m_{nn} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} = \mathbf{M}\mathbf{b}. \tag{2.50}$$

This can also be described as

$$T_\alpha([b_i]) = [\alpha b_i] = \left[\sum_{j=1}^{n} m_{ij} b_j \right]. \tag{2.51}$$

As before, the equation $\alpha\mathbf{b} = \mathbf{M}\mathbf{b}$ is an eigenvalue problem.The solution to the system of equations can often be obtained from the eigenvectors (and eigenvalues), as these consist of basis elements b_i which are monomials in the quotient ring. The matrix $\mathbf{M}$ is called the *action matrix* and α is referred to as the *action monomial/polynomial*.

The action matrix is a transformation matrix which describes how the basis elements are transformed when multiplied by α. It is worth noting that the term transformation matrix often refers to the matrix that explains the mapping of the coordinates rather than the basis elements; in this case that would be $\mathbf{M}^T$. Both the mapping of the basis elements and the coordinates can be of interest, at different situations, and when solving systems of polynomial equations it has turned out to be more convenient to use the former. However, due to this ambiguity other literature might state that the basis vectors are found as the left eigenvectors of the action matrix (then referring to $\mathbf{M}^T$) [21, p. 64],[87, p. 5].

In order to find the matrix $\mathbf{M}$ one can choose the basis $\mathbf{b}$ to be the standard monomials of the quotient ring. Since $\alpha b_i - \sum_{j=1}^{n} m_{ij} b_j = 0 \mod I$ the first term, αb_i can be divided by the Groebner basis to obtain

$$\overline{\alpha b_i}^G = \sum_{j=1}^{n} m_{ij} b_j, \tag{2.52}$$

for all b_i. The remainders after division will be spanned by the standard monomials (that is, $\mathbf{b}$) and $\mathbf{M}$ can be obtained from the coefficients.

However, in practice it can be preferable to avoid division by the Groebner basis. Instead, an *elimination template* can be used to express αb_i in $\mathbf{b}$ without the division [52]. The idea is to find polynomials p_i in the ideal such that

$$p_i = \alpha b_i - \sum_{j=1}^{n} m_{ij} b_j = \sum_{k=1}^{m} h_{ik} f_k \in I, \tag{2.53}$$

where $h_{ik} \in \mathbb{K}$ and f_k are the polynomials generating the ideal. The first thing to do is to take the set of polynomials f_k in the system of equations and multiply them by a number of monomials, in order to get more polynomials and equations. This also requires an extension of the basis vector to contain more monomials. We call the new coefficient matrix $\mathbf{C}$ and the new monomial vector $\mathbf{X}$. Then the extended set of equations can be expressed as $\mathbf{CX} = 0$.

The second thing to do is to partition the monomials in $\mathbf{X}$ into *excessive* monomials $\mathbf{e}$ – those that we want to eliminate; *reducible* monomials that are of the form $\alpha\mathbf{b}$; and basis monomials $\mathbf{b}$. With this, the coefficient matrix will be partitioned into corresponding parts

$$\begin{bmatrix}\mathbf{C}_e & \mathbf{C}_r & \mathbf{C}_b\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \alpha\mathbf{b}\\ \mathbf{b}\end{bmatrix} = 0. \tag{2.54}$$

We then perform Gaussian elimination, corresponding to multiplication from the left by some matrix $\mathbf{A}$, yielding

$$\mathbf{ACX} = \begin{bmatrix}\mathbf{AC}_e & \mathbf{AC}_r & \mathbf{AC}_b\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \alpha\mathbf{b}\\ \mathbf{b}\end{bmatrix} = \begin{bmatrix}0 & \mathbf{I} & -\mathbf{M}\end{bmatrix}\begin{bmatrix}\mathbf{e}\\ \alpha\mathbf{b}\\ \mathbf{b}\end{bmatrix} = 0, \tag{2.55}$$

and the action matrix $\mathbf{M}$ can be found from the multiplication of $\mathbf{A}$ with $\mathbf{C}_b$.

Automatic Generators

This section has briefly described some introductory parts of algebraic geometry and how to solve systems of polynomial equations. When solving large systems there are several other aspects to take into account. In this thesis an automatic generator that generates solution templates for pre-defined problems has been used [53] and the preceding theory is mainly included for completeness. The automatic generator does however, use the methods above. There are also other automatic generators [48]. Despite automatic solver generators being used, the task of parametrising the problem in a solvable way remains.

2.5 Optimisation and Parameter Estimation

As was stated before, if we cannot immediately find an exact solution to a problem or an equation, one way to optimise it can be to find a reasonable solution and then refine it. Many times in computer vision, we want to adjust a number of parameters to best suit a chosen model and the measured data. Different models will be discussed in Chapter 3 and in this section, we will assume that the model is given or has already been chosen. If we

collect the parameters which we want to optimise in $\boldsymbol{\theta}$ and the values that can be measured in $\boldsymbol{\beta}$, we want to minimise the distance from the measured values $\boldsymbol{\beta}_m$ to the by $\boldsymbol{\theta}$ estimated values $\boldsymbol{\beta}_e(\boldsymbol{\theta})$ – where $\boldsymbol{\beta}_e(\boldsymbol{\theta})$ is calculated according to the model. Often, this is expressed as an *error function*

$$E(\boldsymbol{\beta}_m, \boldsymbol{\theta}), \tag{2.56}$$

where E in some way describes how well $\boldsymbol{\beta}_e$ fits to $\boldsymbol{\beta}_m$. This value can also be called the *estimation error*. It is common to use some type of distance between the two values

$$E(\boldsymbol{\beta}_m, \boldsymbol{\theta}) = d(\boldsymbol{\beta}_e(\boldsymbol{\theta}), \boldsymbol{\beta}_m), \tag{2.57}$$

since a small distance shows that the estimation is good. Here, d denotes the distance function. This is also commonly expressed as a function of the difference between the measured and calculated value

$$E(\boldsymbol{\beta}_m, \boldsymbol{\theta}) = f(\boldsymbol{\beta}_e(\boldsymbol{\theta}) - \boldsymbol{\beta}_m). \tag{2.58}$$

One common distance measure is the Euclidean distance. If we denote the different entries of $\boldsymbol{\beta}$ by subscript i, it is defined as

$$d_2(\boldsymbol{\beta}_e(\boldsymbol{\theta}), \boldsymbol{\beta}_m) = ||\boldsymbol{\beta}_e(\boldsymbol{\theta}) - \boldsymbol{\beta}_m||_2 = \sqrt{\sum_i (\beta_{ei}(\theta) - \beta_{mi})^2}. \tag{2.59}$$

Sometimes the squared Euclidean distance is used instead, to avoid the square root. This is also called the l^2*-loss*. If we average the l^2-loss over the number of samples, the *mean squared error* (MSE) and *root mean squared error* (RMSE) are achieved

$$\begin{aligned} \text{MSE} &= \frac{1}{n} \sum_{i=1}^{n} (\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi})^2, \\ \text{RMSE} &= \sqrt{\text{MSE}}. \end{aligned} \tag{2.60}$$

The l^1*-loss* is commonly used as well,

$$d_1(\boldsymbol{\beta}_e(\boldsymbol{\theta}), \boldsymbol{\beta}_m) = ||\boldsymbol{\beta}_e(\theta) - \boldsymbol{\beta}_m||_1 = \sum_i |\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi}|. \tag{2.61}$$

There are also combinations of l^1 and l^2, such as the *Huber loss*, where each term of the loss is quadratic if the value is small, and linear if it is higher

$$d_H(\boldsymbol{\beta}_e(\boldsymbol{\theta}), \boldsymbol{\beta}_m) = \sum_i d_{Hi}(\boldsymbol{\beta}_e(\boldsymbol{\theta}), \boldsymbol{\beta}_m), \tag{2.62}$$

where

$$d_{Hi}(\boldsymbol{\beta}_e(\boldsymbol{\theta}), \boldsymbol{\beta}_m) = \begin{cases} \frac{1}{2}(\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi})^2, & \text{if } |\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi}| \leq \tau, \\ \tau|\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi}| - \frac{1}{2}\tau^2, & \text{otherwise,} \end{cases} \tag{2.63}$$

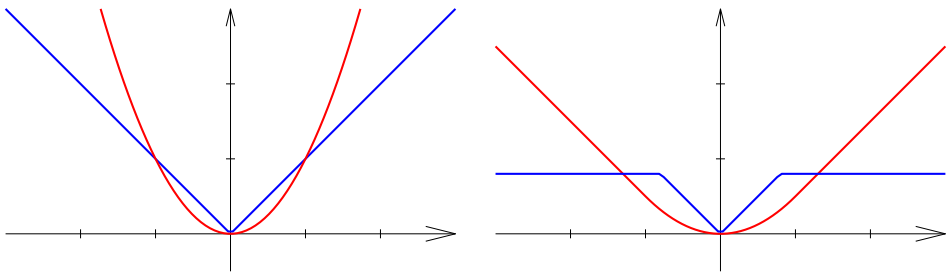


Figure 2.2: In the plots the x-axes show $\boldsymbol{\beta}_e(\boldsymbol{\theta}) - \boldsymbol{\beta}_m$ and the y-axes the error functions $f(\boldsymbol{\beta}_e(\boldsymbol{\theta}) - \boldsymbol{\beta}_m)$. The left plot shows l^1- and l^2-loss in blue and red, respectively. In the right plot the Huber loss with threshold $\tau = 1$ is shown in red and the blue curve shows truncated l^1-loss with threshold $\bar{\tau} = 0.8$.

for some threshold value τ. This threshold value is the limit for when the loss function goes from being quadratic to linear.

Furthermore, there are truncated variants of the loss function. These are similar to the Huber loss in that they use for example l^1 and l^2 for lower parameter values, but then they are constant over a certain threshold $\bar{\tau}$. The *truncated* l^1 *loss* and *truncated* l^2 *loss* are, respectively, defined as

$$\min\left(\sum_i |\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi}|, \bar{\tau}\right) \tag{2.64}$$

$$\min\left(\sum_i (\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi})^2, \bar{\tau}\right). \tag{2.65}$$

Note that $\bar{\tau}$ is a threshold in the total error, while τ in Equation (2.63) is a threshold in the residual. Some of these loss functions, viewed as functions $f(\boldsymbol{\beta}_e(\boldsymbol{\theta}) - \boldsymbol{\beta}_m)$, are plotted in Figure 2.2.

2.5.1 Non-Linear Least Squares

It is common that the measurements $\boldsymbol{\beta}_m$ contain noise, for example from the measurement units. This means that even if some optimal parameters are found, it is unlikely that the error can reach a zero level. Instead, we focus on making it as small as possible. If we return to the error function in Equation (2.59) and denote the *residuals* $r_i(\boldsymbol{\theta}) = \beta_{ei}(\boldsymbol{\theta}) - \beta_{mi}$, the total squared Euclidean distance can be written

$$E(\boldsymbol{\beta}_m, \boldsymbol{\theta}) = ||\boldsymbol{\beta}_m - \boldsymbol{\beta}_e(\boldsymbol{\theta})||_2^2 = \sum_i r_i(\boldsymbol{\theta})^2 = \mathbf{r}^T\mathbf{r}, \tag{2.66}$$

with the *residual vector* $\mathbf{r}(\boldsymbol{\theta})$ containing all values $r_i(\boldsymbol{\theta})$ column stacked. This error is commonly referred to as the sum of squared residuals and to minimise this error by finding the optimal parameters

$$\boldsymbol{\theta}_{\text{opt}} = \text{argmin}_{\boldsymbol{\theta}} E(\boldsymbol{\beta}_m, \boldsymbol{\theta}), \tag{2.67}$$

is known as the *non-linear least squares problem*. Denoting the Jacobian of $\mathbf{r}$ by $\mathbf{J} = \partial \mathbf{r}/\partial \boldsymbol{\theta}$, and differentiating the error, we get

$$\nabla E(\boldsymbol{\theta}) = \frac{\partial E(\boldsymbol{\beta}_m, \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_i 2\nabla r_i(\boldsymbol{\theta})^T r_i(\boldsymbol{\theta}) = 2\mathbf{J}^T\mathbf{r}. \tag{2.68}$$

To minimise this problem we linearise it using second order Taylor expansion

$$E(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) \approx E(\boldsymbol{\theta}) + \left(\frac{\partial E}{\partial \boldsymbol{\theta}}\right)^T \delta\boldsymbol{\theta} + \frac{1}{2}\delta\boldsymbol{\theta}^T \frac{\partial^2 E}{\partial \boldsymbol{\theta}^2}\delta\boldsymbol{\theta}, \tag{2.69}$$

where $\partial^2 E/\partial\boldsymbol{\theta}^2$ denotes the Hessian of E in the point $\boldsymbol{\theta}$. Differentiating this approximation with respect to $\delta\boldsymbol{\theta}$ yields

$$\nabla E(\boldsymbol{\theta} + \delta\boldsymbol{\theta}) = \frac{\partial E}{\partial \boldsymbol{\theta}} + \frac{\partial^2 E}{\partial \boldsymbol{\theta}^2}\delta\boldsymbol{\theta}, \tag{2.70}$$

and by setting this to zero we get the *Newton step*

$$\delta\boldsymbol{\theta} = -\left(\frac{\partial^2 E}{\partial \boldsymbol{\theta}^2}\right)^{-1} \frac{\partial E}{\partial \boldsymbol{\theta}}. \tag{2.71}$$

However, computing the Hessian of E is computationally expensive. By assuming that the residuals r_i are small, we can approximate it according to [10]

$$\frac{\partial^2 E}{\partial \boldsymbol{\theta}^2} = 2\mathbf{J}^T\mathbf{J} + 2\sum_i r_i \nabla^2 r_i \approx 2\mathbf{J}^T\mathbf{J}. \tag{2.72}$$

Then we get

$$\delta\boldsymbol{\theta} = -\left(\frac{\partial^2 E}{\partial \boldsymbol{\theta}^2}\right)^{-1} \nabla E(\boldsymbol{\theta}) = -(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T\mathbf{r}, \tag{2.73}$$

that is, we iteratively update the values of $\boldsymbol{\theta}$ according to

$$\boldsymbol{\theta}^{(k+1)} = \boldsymbol{\theta}^{(k)} - \left(\left(\mathbf{J}^{(k)}\right)^T \mathbf{J}^{(k)}\right)^{-1} \left(\mathbf{J}^{(k)}\right)^T \mathbf{r}^{(k)}, \tag{2.74}$$

where $\mathbf{J}^{(k)}$ and $\mathbf{r}^{(k)}$ are the values of $\mathbf{J}$ and $\mathbf{r}$ for the parameters $\boldsymbol{\theta}^{(k)}$ of iteration k. Using Newton's method on a least squares problem with the approximation above is called the

Gauss-Newton method [10, 32, 55, 92]. Note that in Newton's method, there is a risk that the Hessian is indefinite. However, using the approximation in Equation (2.72) this is not an issue, since $\mathbf{J}^T\mathbf{J}$ is always positive semi-definite.

One problem with the Gauss-Newton method is that there is no guarantee that the step (2.73) is small enough to keep the parameters inside the region where the second order approximation holds. One way to solve this is to use a damped version of the method, for example the *Levenberg-Marquardt*, where the step instead is

$$\delta\boldsymbol{\theta} = -(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{D})^{-1}\mathbf{J}^T\mathbf{r}, \tag{2.75}$$

for some damping parameter λ and some diagonal, positive definite matrix $\mathbf{D}$ [30, 56, 61]. Examples of $\mathbf{D}$ are the identity matrix and a matrix consisting of the diagonal values of $\mathbf{J}^T\mathbf{J}$. Furthermore, the damping parameter can change throughout the optimisation, for example such that it is large in the beginning and smaller later.

Both the Gauss-Newton and the Levenberg-Marquardt methods are iterative, numerical methods that require some stopping criteria. This can for example concern the step size $\delta\boldsymbol{\theta}$ or the size of the residual vector $\mathbf{r}$.

2.5.2 Bundle Adjustment

The term *bundle adjustment* originates from the field of photogrammetry [33], and has later been widely used within computer vision [35]. The name refers to the bundle of light rays that goes from points in space to each camera. The process consists of simultaneous optimisation of the sensors and the structure of the scene. In the case of computer vision, this would refer to the camera positions, their intrinsic parameters and the 3D feature points. The method can also be translated to work for one-dimensional signals, and for sound signals the optimisation would be performed over the senders as well as the receiver positions. In [92], the method is described as: "Bundle adjustment is really just a large sparse geometric parameter estimation problem".

Classically, bundle adjustment is formulated as a non-linear least squares problem as in the previous section. Within this thesis, this is also what has been used, but it is worth mentioning that there does exist a number of variations where the cost models are non-quadratic, see [92]. However, we have mainly been using the following error function

$$E(\boldsymbol{\beta}_m, \boldsymbol{\theta}) = f(\boldsymbol{\beta}_e(\boldsymbol{\theta}) - \boldsymbol{\beta}_m) = \sum_{i=1}^{n} ||\beta_{ei}(\boldsymbol{\theta}) - \beta_{mi}||_2^2, \tag{2.76}$$

or some other variant of the function f with a robust error norm. This is helpful when the measurements $\boldsymbol{\beta}_m$ contain noise, as they mostly do.

For the bundle adjustment to give a good result, a good initialisation is required. For this, the previous section on minimal solvers can be used, but in general this is a research area in itself and will not be covered thoroughly in the introductory part of the thesis. In some of the papers we assume that a good initialisation is already found, while we present a solution for the initialisation in some.

2.6 RANSAC

A big challenge that arises when working with real data is the handling of *outliers*. In the error model in Equation (2.76) each of the n measurements is equally important and if a few measurements are wrong, the penalty for this will be high. Outliers can be measurements that do not fit the model, where the noise is very large. In the presence of outlier values, a parameter estimation that in reality is good can give a large error, which would make us discard that estimation. Actually, many big outliers may prevent us from finding any good solution to the optimisation problem at all. Measurements that fit the model and are not outliers are referred to as *inliers*.

One way to get past this problem is to use the *RANdom SAmpling Consensus* (RANSAC) [29]. The idea with RANSAC is to use as little data as possible to estimate the model parameters and then use the rest of the data to evaluate these parameters. Such estimates can, for example, be found using minimal solvers, which were described in Section 2.4. If m is the minimum number of data points that are needed to estimate parameters for the chosen model, the algorithm works as follows:

Algorithm 2.1: RANSAC

1 **while** *Model is not good enough* **do**
2 Randomly select m data points.
3 Estimate the model parameters $\boldsymbol{\theta}$ from the selected set of points.
4 Count how many of the other data points that are *close enough* (the consensus set) and keep the model if it is the best so far.
5 **end**
6 The model is *good enough*, keep the model and (potentially) improve it using all of the consensus set.

The set of data points that are *close enough* and that are counted in step 4 of Algorithm 1 is called the *consensus set*, and the aim is to get a large consensus set. We have, however, not defined what *close enough* means for the consensus set and how to know that a model is *good enough* to terminate. These are parameters for the RANSAC method that have to be decided. A maximum number of iterations is also required in order to not get stuck when an optimum cannot be found.

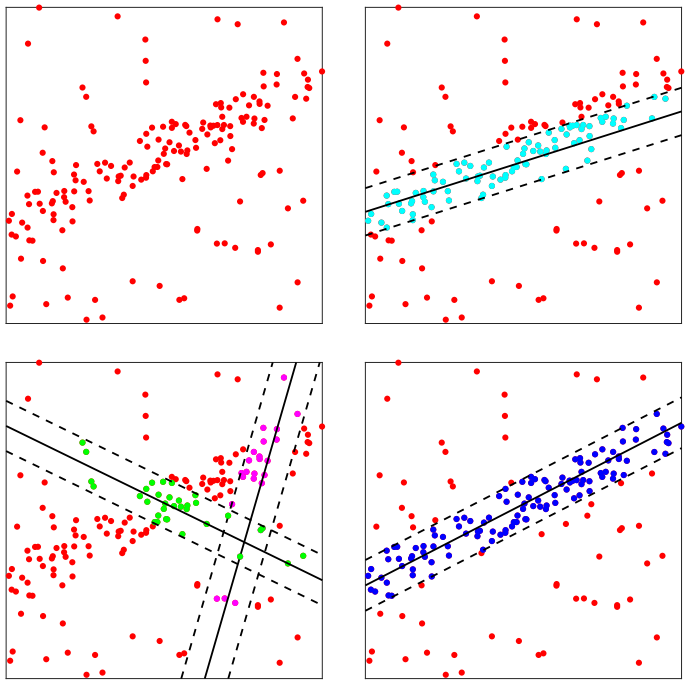


Figure 2.3: The plot on the top left shows a noisy set of points sampled from a line. The plot on the top to the right shows the least squares estimate of the line, given all samples. In the bottom to the left the results from two different RANSAC iterations are shown and the rightmost bottom plot shows the best result after the RANSAC loop is terminated. The solid lines show the estimated lines and the dashed lined show the inlier threshold.

The RANSAC method is specifically good when the set of outliers is large. In this case, RANSAC is robust, can provide a good solution and detect the outliers as well [29]. As mentioned, if the whole dataset is used directly, the outliers might affect the parameter estimation such that the final model does not fit any samples good. The same goes for datasets with missing data. However, since a minimal set of samples is used for the parameter estimation, it is easier to avoid the missing and outlier samples. Finally, the consensus set will give an estimation of which samples that are inliers.

One example of how RANSAC can be used is shown in Figure 2.3. The plot to the top left shows noisy samples from the line $y = 0.5x + 3$. There are also many outlier samples. The task is to, from these samples, estimate the line parameters k and m such that $y = kx + m$. In the rightmost plot on the top row of the figure, the result from a least squares estimation is shown. Due to the many outliers, the line is estimated erroneously as $y = 0.32x + 3.11$. Therefore, we try a RANSAC approach. The minimum number of point that are needed to estimate a line in the plane are two, so in each RANSAC iteration two random points are chosen, whereupon k and m are estimated. The plot in the bottom left of Figure 2.3 shows the results from two such iterations. The green and magenta dots are the consensus

set for the two lines, respectively. The dashed lines show the limit for when points are inliers, and the distance from the solid line to the dashed is the threshold. The plot to the bottom right shows the best result after the RANSAC loop is terminated, which was the line $y = 0.51x + 3.1$.

Since the first paper about RANSAC was published, a number of variations have been presented and today these are also widely used. For some of them, see [17, 43, 73]. These different RANSAC methods are good for finding an initial estimation of the parameters. Once the RANSAC loop is terminated, it is common that some local optimisation is performed over the inlier set.

Using RANSAC followed by l^2-optimisation over the inliers can be compared to optimisation using truncated l^2 loss. When we remove the outliers using RANSAC, we decide that values for which the residual is larger than some threshold should not be penalised as hard in the total error as those with smaller residuals. If we use the same threshold in truncated l^2 we get the same effect. Therefore, in one way RANSAC followed by l^2-optimisation can be seen as a heuristic for achieving truncated l^2-optimisation.

Chapter 3

Sensor Modelling

In this chapter details about sensor modelling are presented and the analysis of signals is developed. First, the focus will be on sound and then we also present modelling equations for images.

3.1 Sensor Modelling for Sound

This section will present models for sound, but the same principles often apply to other one-dimensional signals, such as UWB. Firstly, we will focus on measurements that can be used to calculate sender and receiver positions and thus can be used for localisation and mapping.

3.1.1 TOA and TDOA

Assume that we have a setup with a number of receivers $\mathbf{r}_i \in \mathbb{R}^3$, $i = 1, .., m$ and senders $\mathbf{s}_j \in \mathbb{R}^3$, $j = 1, .., n$. Also, assume that these are synchronised, that is, that their internal clocks coincide. Then, by comparing when the signals were emitted from the senders to when they reached the receivers, we can get the travel time, and by multiplying these measurements with the speed of the signal v, the absolute distance measures between each sender and receiver can be derived. The distance d_{ij} will be

$$d_{ij} = v(t_{ij} - T_j) = \|\mathbf{r}_i - \mathbf{s}_j\|, \tag{3.1}$$

where t_{ij} denotes the arrival time for signal j to receiver i and T_j is the emission time. This time difference is called the *time of arrival* (TOA) measurement or absolute travel time. All

distances d_{ij} can be collected in a *distance matrix*

$$\mathbf{D} = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{m1} & d_{m2} & \dots & d_{mn} \end{bmatrix}. \tag{3.2}$$

One could also consider a setup where the receivers are synchronised, but not the senders. That would instead give us *time-difference of arrival* (TDOA) measurements, or the relative travel time; for emitted sound, we know how much longer it took for the sound to reach receiver 2 compared to receiver 1, etc. This could be explained by the following equation

$$z_{ikj} = \|\mathbf{r}_i - \mathbf{s}_j\| + o_j. \tag{3.3}$$

Here, z_{ikj} is the TDOA value and o_j is an offset that is different for each sound event and the arrival and emission times are not synchronised. The explanation of the third index k becomes evident when we look at the next equation. What we actually measure in this case is

$$(t_{ij} - T_j) - (t_{kj} - T_j) = t_{ij} - t_{kj}, \tag{3.4}$$

that is, how much longer it took for sound j to reach receiver i than receiver k. The value can be multiplied by v to give the difference in distance. Equation (3.3) can be obtained from Equation (3.1),

$$\|\mathbf{r}_i - \mathbf{s}_j\| = v(t_{ij} - T_j) = vt_{ij} - vT_j. \tag{3.5}$$

Now, we add and subtract the constant $T_0 = t_{kj}$ from this equation, which gives

$$\|\mathbf{r}_i - \mathbf{s}_j\| = v(t_{ij} - T_0 + T_0 - T_j) = v(t_{ij} - t_{kj}) + v(T_0 - T_j). \tag{3.6}$$

Note that the receiver index k in t_{kj} is fixed. Since v is known and $t_{ij} - t_{kj}$ can be measured, we call that term $z_{ikj} = v(t_{ij} - t_{kj})$, while the part that cannot be measured will be the offset $o_j = -v(T_0 - T_j)$, which results in Equation (3.3). The value z_{ikj} ca also be expressed as

$$z_{ikj} = \|\mathbf{r}_i - \mathbf{s}_j\| - \|\mathbf{r}_k - \mathbf{s}_j\|. \tag{3.7}$$

For an illustration of TOA and TDOA, see Figure 3.1.

There are also other variations of TOA and TDOA. One can think of a situation where the senders are not synchronised with the receivers, but where the signals are emitted regularly. This would give an expression similar to Equation (3.3), but with a constant offset, that is

$$z_{ikj} = \|\mathbf{r}_i - \mathbf{s}_j\| + o. \tag{3.8}$$

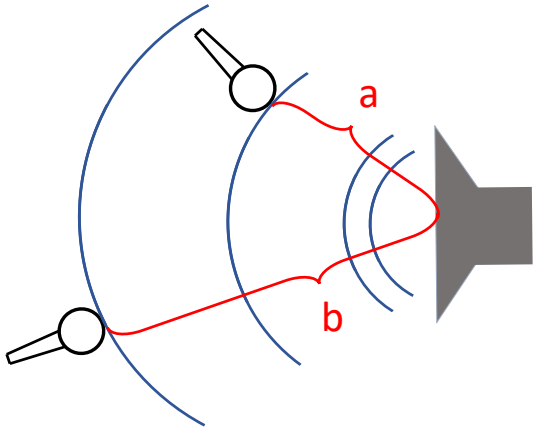


Figure 3.1: The figure is illustrating TOA and TDOA measures. The sound emitted from the speaker will reach the two microphones at different times. For TOA we have that $d_{11} = a$ and $d_{21} = b$. For TDOA we instead measure the difference, $(t_{21} - T_1) - (t_{11} - T_1) = (b - a)/v$.

This case will be referred to as *constant offset time-difference of arrival* (COTDOA). Here, the emission times can be expressed as $T_j = T_1 + (j-1) \cdot \phi$, if ϕ is the time between the sound events.

Furthermore, if neither the senders nor the receivers are synchronised, we get the *unsynchronised time-difference of arrival* (UTDOA) problem. This equation will also be similar to Equation (3.3), but with a second constant, connected to the receivers

$$z_{ikj} = \|\mathbf{r}_i - \mathbf{s}_j\| + q_i + o_j. \tag{3.9}$$

Papers I, IV and V in this thesis uses regular TDOA measurements, while Paper II focuses on the COTDOA problem and the methods in Paper III works for either. Note that by subtracting the offsets o_j or o (together with q_i) from z_{ikj} we get the distance measurements d_{ij}.

The TDOA Vector and Matrix

In the section above, we let one of the receivers be fixed. There are also cases when the sender is fixed, while there are several receivers. Then, the index j will instead be constant, while we vary i and k. In this case, the different TDOA measurements in Equation 3.7 can

be collected in an $m \times m$ (m being the number of receivers) matrix

$$\mathbf{Z} = \begin{bmatrix} z_{11j} & \cdots & z_{1mj} \\ \vdots & \ddots & \vdots \\ z_{m1j} & \cdots & z_{mmj} \end{bmatrix}. \tag{3.10}$$

We will refer to $\mathbf{Z}$ as the *TDOA matrix*. Each of the columns in $\mathbf{Z}$ can be used as a *TDOA vector*

$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & \dots v_m \end{bmatrix}^T = \begin{bmatrix} z_{1kj} & z_{2kj} & \dots z_{mkj} \end{bmatrix}^T. \tag{3.11}$$

This corresponds to taking the relative distances from one receiver to all the others. Note that we can express the TDOA matrix as $\mathbf{Z} = \mathbf{v}\mathbf{1}^T - \mathbf{1}\mathbf{v}^T$, where $\mathbf{1}$ is a vector with all entries equal to 1. The TDOA matrix formulation has, for example, been used in [5, 94].

It is worth mentioning that the indexing of the TDOA measurements might vary depending on the use-case. If one of the receivers is fixed, the measurements from Equation 3.7 are often denoted

$$z_{ij} = z_{ikj} = ||\mathbf{r}_i - \mathbf{s}_j|| - ||\mathbf{r}_k - \mathbf{s}_j||. \tag{3.12}$$

This is the case in Papers I–IV. However, if the sender is constant, the measurements from Equation 3.7 might instead be expressed as

$$z_{ik} = z_{ikj} = ||\mathbf{r}_i - \mathbf{s}_j|| - ||\mathbf{r}_k - \mathbf{s}_j||. \tag{3.13}$$

This notation has been used in Paper V. If only two subscripts are used the meaning of z_{ab} has to be taken from the context.

3.1.2 Model Selection and Parameter Estimation

To find the optimal model parameters from the expression in Equation (2.67) the model must first be decided. Assume that we have two measurements from different receivers, where the received signals $\mathbf{x}(n)$ and $\bar{\mathbf{x}}(n)$ come from the same emitted signal. The easiest connection between these two is to consider one of them to be a translated version of the other,

$$\mathbf{x}(n) = \bar{\mathbf{x}}(n + h). \tag{3.14}$$

This corresponds to, for example, TDOA, where h would be the time difference value in Equation (3.4). If $\bar{\mathbf{x}}(n)$ rather is the emitted signal while $\mathbf{x}(n)$ is the received one, h corresponds to the TOA value. Going back to them both being received signals, the connection between the them could also look differently. For example, it is reasonable to assume that the signal is stronger when it reaches a receiver that is close to the sender, compared to one that is further away. Therefore, an amplitude parameter γ might be added

$$\mathbf{x}(n) = \gamma\bar{\mathbf{x}}(n + h). \tag{3.15}$$

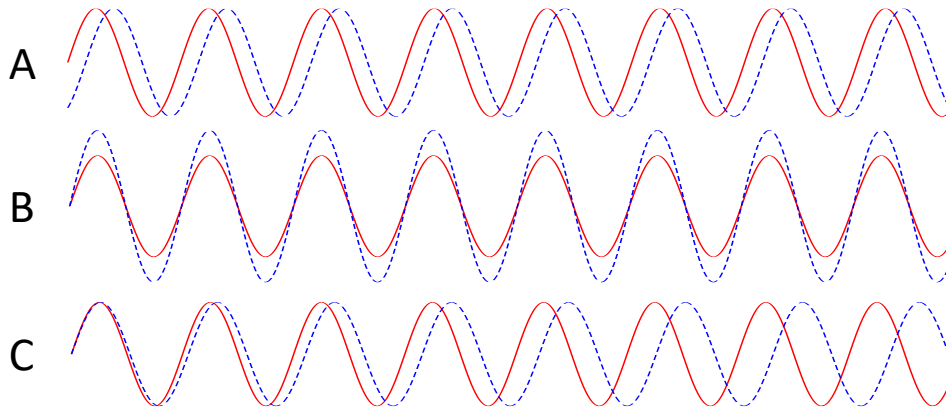


Figure 3.2: This shows what different sound models can capture. In A we have a sound and a translated copy of it, connected to Equation (3.14). Plot B shows what happens if we only have the impact of an amplitude, corresponding to γ, with $h = 0$, in Equation (3.15) and C shows the same thing but for Doppler, corresponding to $h = 0$, $\gamma = 1$ in Equation (3.16).

Furthermore, if the sender is moving while emitting the sound this may result in a stretched or compressed signal. This raises the need of a Doppler parameter α,

$$\mathbf{x}(n) = \gamma\bar{\mathbf{x}}(\alpha n + h). \tag{3.16}$$

The effect of a translation, an amplitude difference and the presence of a Doppler factor is shown in Figure 3.2. The models in Equations (3.14), (3.15) and (3.16) have been used in Paper I. One could also think of a number of other suitable models, and the model might need to be adjusted to the specific problem.

These models could be used to express an optimisation problem. If we collect the parameters of interest in $\boldsymbol{\theta}$. For the model in (3.14) we would have $\boldsymbol{\theta} = \{h\}$ and for the model in (3.16) we have $\boldsymbol{\theta} = \{h, \gamma, \alpha\}$, while $\boldsymbol{\beta} = \{\mathbf{x}, \bar{\mathbf{x}}\}$ in both cases.

3.1.3 Estimating TOA and TDOA

Once the model has been decided, the error function from Equation (2.76) can be used to estimate the parameters in $\boldsymbol{\theta}$. Again, assume that we have two signals $\mathbf{x}$ and $\bar{\mathbf{x}}$ and that we want to find how these relate to one another. In this case, we can express the error function as

$$f(\boldsymbol{\beta}, \boldsymbol{\theta}) = \sum_n (\mathbf{x}(n) - (\eta(\bar{\mathbf{x}}, \boldsymbol{\theta})))^2, \tag{3.17}$$

where $\boldsymbol{\beta} = \{\mathbf{x}, \bar{\mathbf{x}}\}$ and $\eta(\bar{\mathbf{x}}, \boldsymbol{\theta})$ can be for example $\bar{\mathbf{x}}(n+h)$ or $\gamma\bar{\mathbf{x}}(\alpha n+h)$, in accordance with the previous section. We see that the distance function d is given by the squared difference between the two signals. Once this error function is formulated, the parameter estimation can be found using the methods previously described, in Section 2.5.2.

As mentioned above, if $\mathbf{x}$ is the emitted signal, $\bar{\mathbf{x}}$ is the received signal and the model is that of Equation (3.14) the estimated value h will represent the time of arrival. If we instead choose $\mathbf{x}$ and $\bar{\mathbf{x}}$ to be the signals received by two different receivers coming from the same emitted signal, we will estimate a time-difference of arrival.

Furthermore, if we use the model in Equation (3.14) we can also estimate the parameter h using cross-correlation. Remember the cross-correlation for two real signals $\mathbf{x}$ and $\bar{\mathbf{x}}$ from Equation (2.20)

$$(\mathbf{x} \star \bar{\mathbf{x}})(h) = \sum_n \mathbf{x}(n)\bar{\mathbf{x}}(n+h). \tag{3.18}$$

The translation h is obtained by maximising the cross-correlation function,

$$h_{opt} = \text{argmax}_h(\mathbf{x} \star \bar{\mathbf{x}})(h). \tag{3.19}$$

This estimation will be exactly the same as the one coming from minimisation of (3.17), since

$$\begin{aligned}\text{argmin}_h f(\boldsymbol{\beta}, \boldsymbol{\theta}) = \text{argmin}_h \sum_n (\mathbf{x}(n) - \bar{\mathbf{x}}(n+h))^2 = \text{argmin}_h \sum_n \Big((\mathbf{x}(n))^2 + \\ (\bar{\mathbf{x}}(n+h))^2 - 2\mathbf{x}(n)\bar{\mathbf{x}}(n+h) \Big) = \text{argmin}_h \sum_n -2\mathbf{x}(n)\bar{\mathbf{x}}(n+h) \\ = \text{argmax}_h \sum_n \mathbf{x}(n)\bar{\mathbf{x}}(n+h) = \text{argmax}_h (\mathbf{x} \star \bar{\mathbf{x}})(h).\end{aligned} \tag{3.20}$$

Hence, for the models in Equations (3.15) and (3.16) we use error function (3.17), but if we stay with the smaller model (3.14) (that is if $\gamma = 1$ and $\alpha = 1$) we might as well use cross-correlation to find the h that minimises the error function.

There are also variants of cross-correlation, for example generalised cross-correlation with phase transform (GCC-PHAT), which we used for initialisation in Paper I. In GCC-PHAT, the correlation value is scaled by the magnitude of the spectrum [42]. If we let $\boldsymbol{\mathcal{X}}(f)$ and $\bar{\boldsymbol{\mathcal{X}}}(f)$ be the Fourier transform of $\mathbf{x}(n)$ and $\bar{\mathbf{x}}(n)$, respectively, the Fourier transform of the cross correlation will be

$$\mathcal{F}(\mathbf{x} \star \bar{\mathbf{x}}) = \boldsymbol{\mathcal{X}}(f) \cdot \overline{\bar{\boldsymbol{\mathcal{X}}}(f)}, \tag{3.21}$$

where $\overline{\bar{\boldsymbol{\mathcal{X}}}(f)}$ represents the complex conjugate of $\bar{\boldsymbol{\mathcal{X}}}(f)$. Then, the GCC-PHAT is given by

$$\text{GCC-PHAT}_{\mathbf{x},\bar{\mathbf{x}}} = \mathcal{F}^{-1}\left(\frac{\boldsymbol{\mathcal{X}}(f) \cdot \overline{\bar{\boldsymbol{\mathcal{X}}}(f)}}{\left|\boldsymbol{\mathcal{X}}(f) \cdot \overline{\bar{\boldsymbol{\mathcal{X}}}(f)}\right|} \right), \tag{3.22}$$

where the symbol $\mathcal{F}^{-1}$ represents the inverse Fourier transform. With this, GCC-PHAT can be used in Equation 3.19 instead of the regular cross-correlation function. By this, all frequencies get an equal weighing. This increases the robustness against reverberation. For more information, see [13, 42].

There are also cases where the time delay or time-difference of arrival are easier to find. If the signal consists of distinct sound events, for example claps, this will give a clear peak in the signal. The value of h can then be found by simply detecting the first or largest peak in both signals and differentiate the time/sample number of these.

Estimation on Subsample Level

In the equations above, the translation h will be estimated as an integer number of samples. To refine the estimations further, we can do the parameter estimation on continuous signals, achieved from ideal interpolation. This would result in the following error function

$$f(\boldsymbol{\beta}_m, \boldsymbol{\theta}) = \int_t (\mathbf{x}_a(t) - \tau(\bar{\mathbf{x}}_a, \boldsymbol{\theta}))^2 \, \mathrm{d}t, \tag{3.23}$$

with τ defined in accordance with η above, but for continuous values (remember that $\tau(\bar{\mathbf{x}}_a, \boldsymbol{\theta})$ in either case contains a t which is is integrated over). Minimising this error would result in parameters on an even finer scale than the sample rate, and thus even more exact measures. However, the estimation also becomes more sensitive to noise. This matter is considered in Paper I.

3.2 Sensor Modelling for Vision

For computer vision and 3D reconstructions the pinhole camera model is a common way to model the relation between the 3D points and 2D points matrices [35]. Two alternatives are the affine and projective camera models. We will begin by introducing the pinhole camera.

3.2.1 Camera Models

The *pinhole camera* model takes a point $\hat{\mathbf{U}} = \begin{bmatrix} X & Y & Z \end{bmatrix}^T \in \mathbb{R}^3$ in 3D to a point $\hat{\mathbf{u}} = \begin{bmatrix} x & y \end{bmatrix}^T \in \mathbb{R}^2$ in the image by following the straight line from $\hat{\mathbf{U}}$ to the camera centre in the origin, $\hat{\mathbf{C}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$, and intersecting it with the image plane. If the distance from the camera centre to the image plane is 1 and we assume that the image plane is parallel to

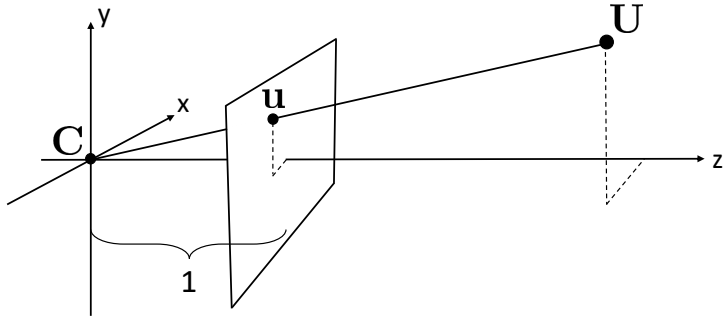


Figure 3.3: An illustration of the pinhole camera model. The camera centre $\mathbf{C}$ is located in the origin in the xyz-coordinate system. The image plane is parallel to the xy-plane and located at unit distance from $\mathbf{C}$ in the positive z-direction. A 3D point $\mathbf{U}$ is mapped to the image point $\mathbf{u}$, where $\mathbf{u}$ is given by the intersection of the image plane and the straight line going from $\mathbf{U}$ to $\mathbf{C}$, see Equation (3.24).

the xy-plane – that is, the viewing direction is along the z-axis – the projection is given by

$$\hat{\mathbf{u}} = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} X/Z \\ Y/Z \end{bmatrix}, \tag{3.24}$$

see Figure 3.3.

For camera modelling it is convenient to describe the points in 2D and 3D using homogeneous coordinates, that is, we write $\hat{\mathbf{u}}$ as $\mathbf{u} = \begin{bmatrix} x & y & 1 \end{bmatrix}^T \in \mathcal{P}^2$ and represent $\hat{\mathbf{U}}$ by $\mathbf{U} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T \in \mathcal{P}^3$. The symbol $\mathcal{P}$ represents the *projective space* [90]. For a general camera projection matrix $\mathbf{P} \in \mathbb{R}^{3\times 4}$, we have the relationship

$$\begin{bmatrix} a \\ b \\ c \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{3.25}$$

where we then can divide by c to obtain the image point in Cartesian coordinates, $\begin{bmatrix} a/c & b/c & 1 \end{bmatrix}^T = \begin{bmatrix} x & y & 1 \end{bmatrix}^T$. We usually write this as a proportionality,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \sim \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.26}$$

Any point $\lambda \begin{bmatrix} x & y & 1 \end{bmatrix}^T \in \mathcal{P}^2$, for any value of λ, represents the same image point since $\lambda \begin{bmatrix} x & y & 1 \end{bmatrix}^T \sim \begin{bmatrix} x & y & 1 \end{bmatrix}^T$. The same is true for points in $\mathcal{P}^3$, multiplication with a constant does not change which 3D points they represent.

Also lines can be described in the projective space, similar to points. A line in $\mathbb{R}^2$ with the equation

$$\alpha x + \beta y + \gamma = 0, \tag{3.27}$$

can be represented as $\mathbf{l} = \begin{bmatrix}\alpha & \beta & \gamma\end{bmatrix}^T \in \mathcal{P}^2$. Through this, we can also determine if a point $\mathbf{x} = \begin{bmatrix}\lambda x & \lambda y & \lambda\end{bmatrix}^T \in \mathcal{P}^2$ lies on the line by looking at the scalar product of the vectors

$$\mathbf{x}^T\mathbf{l} = \begin{bmatrix}\lambda x & \lambda y & \lambda\end{bmatrix} \begin{bmatrix}\alpha \\ \beta \\ \gamma\end{bmatrix} = \lambda(\alpha x + \beta y + \gamma). \tag{3.28}$$

If this is equal to zero, the point lies on the line.

We now return to the representation of a camera $\mathbf{P}$. For the pinhole camera model the camera matrix is $\mathbf{P} = \begin{bmatrix}\mathbf{I} & \mathbf{0}\end{bmatrix}$, with $\mathbf{I}$ being the 3×3 identity matrix, and $\mathbf{0}$ a 3×1 vector of zeroes. If the camera moves, that can be represented by a 3×3 rotation matrix $\mathbf{R}$ (that is, $\mathbf{R}^T\mathbf{R} = \mathbf{I}$ and $\det(\mathbf{R}) = 1$) and a 3×1 translation vector $\mathbf{t}$, giving $\mathbf{P} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$. These are sometimes referred to as *extrinsic parameters*.

A more general model of the camera matrix is

$$\mathbf{P} = \mathbf{K}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} = \begin{bmatrix}\mu f & s & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1\end{bmatrix} \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}. \tag{3.29}$$

The matrix $\mathbf{K}$ describes the *intrinsic parameters* of the camera. The value f is called the focal length and is a re-scaling parameter that changes the image coordinates to pixels. Both the aspect ratio μ and the skew parameter s re-scales for non-square pixels (and are often one and zero, respectively) and the point $\begin{bmatrix}x_0 & y_0\end{bmatrix}^T$ is referred to as the principal point, and gives a translation to get the correct image centre. The effect of some of these parameters is shown in Figure 3.4. A camera represented by the matrix in Equation (3.29) is called a *finite projective camera*. Such cameras have eleven degrees of freedom [35].

We call $\mathbf{P}$ and affine camera if it has the structure

$$\mathbf{P} = \begin{bmatrix}\mathbf{A} & \mathbf{t} \\ \mathbf{0} & 1\end{bmatrix}, \tag{3.30}$$

where $\mathbf{A}$ is a 2×3 matrix, $\mathbf{t}$ has size 2×1 and $\mathbf{0}$ is 1×3 vector of zeroes. The affine camera is a good approximation when the distance from the camera to each point in the scene is approximately the same. A *general projective camera* is given by an arbitrary 3×4 matrix with rank 3 [35]. The affine camera has eight degrees of freedom, while the general projective camera has eleven. In this thesis, most cameras will be assumed to be finite projective cameras.

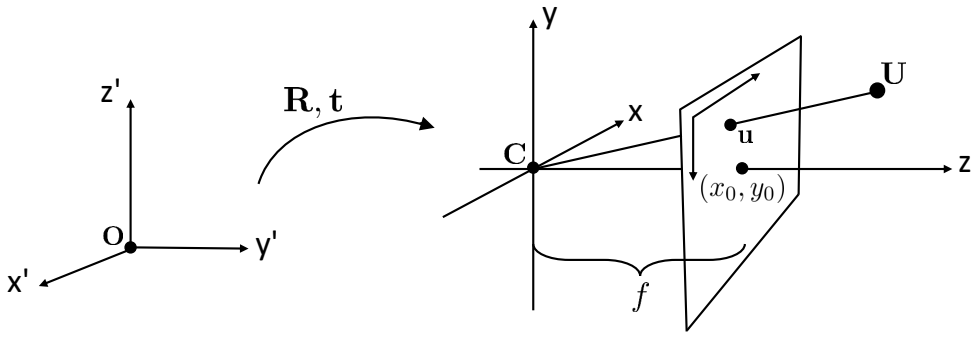


Figure 3.4: An illustration of the extrinsic parameters $\mathbf{R}$ and $\mathbf{t}$ and the intrinsic parameters f, x_0 and y_0. The matrix $[\mathbf{R} \;\; \mathbf{t}]$ represents a change in coordinate system from the world coordinates to one where the camera centre is placed in the origin. The focal length f is the distance from the camera centre to the image plane and the principal point $[x_0 \;\; y_0]^T$ gives a translation to the correct image centre.

3.2.2 Camera Calibration

The matrix $\mathbf{K}$ in Equation (3.29) corresponds to a transformation of the points in the image plane in $\mathbb{R}^2$ to the actual image coordinates, measured in pixels, while the matrix $\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ transforms the position and coordinate system of the camera. If the intrinsic parameters – that is, the values in $\mathbf{K}$ – are known, the camera $\mathbf{P}$ is said to be calibrated, and $\mathbf{K}$ is referred to as the *calibration matrix* [35]. By applying the inverse of the calibration matrix to Equation (3.26) the normalised coordinates,

$$\mathbf{u}' = \mathbf{K}^{-1}\mathbf{u} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}\mathbf{U}, \tag{3.31}$$

are obtained. Similarly, $\mathbf{K}^{-1}\mathbf{P} = \begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}$ is said to be a normalised camera. How to find the camera matrix $\mathbf{P}$ and the calibration matrix $\mathbf{K}$ will be discussed further in Section 4.3.

3.2.3 Parameter Estimation for Vision

When it comes to vision, we can always measure the images, and thus the image points, while the 3D points and the camera matrices might be known or unknown. Assume that we have a set of m cameras which each capture n different 3D points. If both 3D points and cameras are unknown, our sought set of parameters will be $\boldsymbol{\theta} = \{\mathbf{P}_1, ..., \mathbf{P}_m, \mathbf{U}_1, ..., \mathbf{U}_n\}$. The measurable quantities will be the image points that come from projecting each 3D point in each camera, $\boldsymbol{\beta}_m = \{\mathbf{u}_{11}, ..., \mathbf{u}_{1n}, \mathbf{u}_{21}, ..., \mathbf{u}_{mn}\}$, where $\mathbf{u}_{ij} \sim \mathbf{P}_i\mathbf{U}_j$ and $\boldsymbol{\beta}_m$ in total contains $2mn$ values, since each $\mathbf{u}_{ij}$ has two unknown coordinates. If the 3D points or the cameras are known, we move these parameters from $\boldsymbol{\theta}$ to $\boldsymbol{\beta}_m$.

Chapter 4

Mapping and Localisation

This chapter will discuss the subjects of mapping and localisation, both using TOA/TDOA data and images. Before these terms are explained, the following is worth noting: In this thesis we treat, for example, sound and images as different types of signals, but we also focus on the similarities in how they can be used. In many cases, such as the map merging treated later in Chapter 5, it is of little importance which type of signal that was used to create the map. Therefore, this chapter includes both mapping and localisation, using both sound and vision.

Locating the position of several *anchors* or *nodes* for one-dimensional signals, for example receivers, is a type of *mapping*, where we use measurements and a priori information (in this case about the position of the sender) to create a map of the setup. If we are to find the position of the sender among a number of known receivers, this can also be referred to as *localisation*, as we localise in the previously mapped environment. In a similar way, we can do mapping of a scene using a camera with known intrinsic and extrinsic parameters, as we find the position of objects in the scene. Once this is done, this known map can be used for localisation in the scene, that is, to find an unknown camera position.

As explained above, the mapping and localisation steps are performed separately, but given enough sensor data both parts can actually occur at once. This is often referred to as *simultaneous localisation and mapping* (SLAM) and *structure from motion* (SfM) for images. Using sound, this is sometimes called *structure from sound*, because of the similarities to SfM. Simultaneous localisation and mapping can also be referred to as the *calibration problem* for one-dimensional signals.

Before we go into the details on the algorithms, we want to mention a few things about noise. As mentioned in Section 2.5.1 and 2.5.2, the measurement values – either TOA/TDOA values or image point positions – often contain noise. This makes it impossible

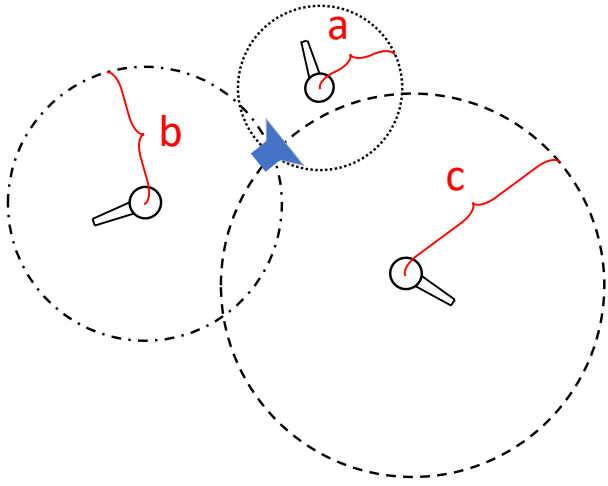


Figure 4.1: The figure shows trilateration in a plane using TOA measurements from one sound source and three receivers. The location of the speaker is unknown and the positions of the microphones, $\mathbf{r}_1$, $\mathbf{r}_2$, $\mathbf{r}_3$ are known. Furthermore, the distances $d_{11} = a$, $d_{21} = b$ and $d_{31} = c$ are measured. The location of the loudspeaker is given by the intersection of the three circles.

to find a perfect solution and that has to be taken into account when the parameters are estimated, for example using least squares. However, when we solve minimal cases (see Section 2.4) we will assume that there is no noise, and solve the equations exactly – in many cases to get an initial solution which can then be improved on. For this reason, some of the measurement equations below will contain noise, while some will not. As this might seem a bit ambiguous at first sight, it is important to read the equations with their context in mind.

4.1 Trilateration and Multilateration

Trilateration is a process for determining the position of a sender given the distances to several receivers, that is TOA distances. To illustrate the idea, we consider two receivers $\mathbf{r}_1$, $\mathbf{r}_2$ and one sender $\mathbf{s}_1$, all located in a plane. For these we have TOA measurements, such that the distances d_{11} and d_{21} from the sender to each of the receivers are known. Then there are two potential points where the sender could be located, namely where the circle with centre in $\mathbf{r}_1$ and radius d_{11} and the circle with centre in $\mathbf{r}_2$ and radius d_{21} intersect. If we add a third receiver $\mathbf{r}_3$, the circle around that with radius d_{31} will contribute to a single intersection point of the three circles (assuming that they do not lie on a line) and this will be the solution for the position of $\mathbf{s}_1$. For an illustration, see Figure 4.1.

In three dimensions each distance measure d_{ij} defines the radius of a sphere with $\mathbf{r}_j$ as centre,

and in a similar way the positions of the senders can be found as the intersections of the different spheres. If we have few measurements, we might get several possible solutions. The opposite problem, when senders are known and receivers and unknown, is solved identically if we have TOA measurements – that is, it does not matter for the algorithm whether a node or an anchor is a sender or a receiver.

The corresponding problem for TDOA measurements is called multilateration. The difference here is that we also have to estimate an offset for each sender. The principles are, however, similar, but instead of knowing the actual radii, we know the differences between the radii. Therefore, the solution will be given by the intersection of a number of hyperboloids rather than spheres.

In the presence of noise there will not be an actual intersection of all curves or surfaces and the system has to be be solved in a least squares sense.

4.2 Self-Calibration for TOA and TDOA

From the TOA or TDOA measurements we can formulate a large system of equations, representing the spheres or hyperboloids that were discussed in the previous section. We could have a static setup where the receiver positions are known and the sender positions are to be found, as in the small 2D example above. The problem could also be the opposite – that the sender positions are known while the receiver positions are unknown. As stated before, given enough measurements, it is also possible to calculate both receiver and sender positions, if all are unknown [53, 91]. This is called *self-calibration* or *automatic calibration*.

4.2.1 Systems of Equations

If we again let j denote the sender number and i the number of the receiver, squaring Equation (3.1) for the TOA case will yield the system of equations

$$d_{ij}^2 = \|\mathbf{r}_i - \mathbf{s}_j\|^2 = (\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{s}_j), \tag{4.1}$$

and in the case of TDOA, re-arranging and squaring Equation (3.3) we get

$$(z_{ij} - o_j)^2 = \|\mathbf{r}_i - \mathbf{s}_j\|^2 = (\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{s}_j), \tag{4.2}$$

for all i and j. If we have measurements of different values d_{ij} or z_{ij}, they probably contain noise and we rather get the models

$$\tilde{d}_{ij} = \sqrt{(\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{s}_j)} + \epsilon_{ij}, \tag{4.3}$$

$$\tilde{z}_{ij} = \sqrt{(\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{s}_j)} + o_j + \varepsilon_{ij}, \tag{4.4}$$

where ϵ_{ij} and ε_{ij} are noise terms representing either inliers – assumed to be normally distributed with a relatively small standard deviation – or outliers – assumed to be drawn from a uniform distribution with significantly larger standard deviation.

Independently of whether there is noise or not, if we have m receivers and n senders this will result in $m \cdot n$ equations to be solved. An initial guess for the unknowns can be found using the methods described in Section 2.4. However, the problem first needs to be formulated correctly. This can be done in several ways, of which a few will be mentioned here.

Calibration Using TOA

We assume that both sender and receiver positions are unknown and that we have TOA measurements between all senders and receivers. The following procedure to solve for the positions was presented in [47]. First, looking at Equation (4.1), this can be expanded to

$$d_{ij}^2 = (\mathbf{r}_i - \mathbf{s}_j)^T(\mathbf{r}_i - \mathbf{s}_j) = \mathbf{r}_i^T\mathbf{r}_i + \mathbf{s}_j^T\mathbf{s}_j - 2\mathbf{r}_i^T\mathbf{s}_j. \tag{4.5}$$

Organising all the distances d_{ij} in a matrix $\mathbf{D}$, as in Equation (3.2), followed by element-wise squaring (denoted by superscript $\circ 2$) gives

$$\mathbf{D}^{\circ 2} = \begin{bmatrix} d_{11}^2 & d_{12}^2 & \dots & d_{1n}^2 \\ d_{21}^2 & \ddots & & d_{2n}^2 \\ \vdots & & \ddots & \vdots \\ d_{m1}^2 & d_{m2}^2 & \dots & d_{mn}^2 \end{bmatrix}. \tag{4.6}$$

By subtracting the first column of $\mathbf{D}^{\circ 2}$ from the rest of the columns and thereafter subtracting the first row from the rest of the rows we obtain

$$\mathbf{B} = \begin{bmatrix} d_{11}^2 & d_{12}^2 - d_{11}^2 & \dots & d_{1n}^2 - d_{11}^2 \\ d_{21}^2 - d_{11}^2 & & & \\ \vdots & & \bar{\mathbf{B}} & \\ d_{m1}^2 - d_{11}^2 & & & \end{bmatrix}, \tag{4.7}$$

where $\bar{\mathbf{B}}$ is an $(m-1) \times (n-1)$ matrix with entries $d_{ij}^2 - d_{i1}^2 - d_{1j}^2 + d_{11}^2$, for $i = 2, \dots, m$ and $j = 2, \dots, n$. This matrix is called the *double compaction matrix* [7, 46]. Doing the corresponding operations to the matrix with entries $\mathbf{r}_i^T\mathbf{r}_i + \mathbf{s}_j^T\mathbf{s}_j - 2\mathbf{r}_i^T\mathbf{s}_j$ results in a system of equations that is equivalent to the original system in Equation (4.5). The row and column operations cancel some of the quadratic terms in $\mathbf{r}_i$ and $\mathbf{s}_j$, leaving us with a system with four types of equations:

1. one equation $d_{11}^2 = \mathbf{r}_1^T\mathbf{r}_1 + \mathbf{s}_1^T\mathbf{s}_1 - 2\mathbf{r}_1^T\mathbf{s}_1$,

2. $n-1$ equations $d_{1j}^2 - d_{11}^2 = \mathbf{s}_j^T\mathbf{s}_j - \mathbf{s}_1^T\mathbf{s}_1 - 2\mathbf{r}_1^T(\mathbf{s}_j + \mathbf{s}_1)$,
3. $m-1$ equations $d_{i1}^2 - d_{11}^2 = \mathbf{r}_i^T\mathbf{r}_i - \mathbf{r}_1^T\mathbf{r}_1 - 2(\mathbf{r}_i - \mathbf{r}_1)^T\mathbf{s}_1$,
4. $(m-1)(n-1)$ equations $d_{ij}^2 - d_{i1}^2 - d_{1j}^2 + d_{11}^2 = -2(\mathbf{r}_1^T\mathbf{s}_1 + \mathbf{r}_i^T\mathbf{s}_j - \mathbf{r}_i^T\mathbf{s}_1 - \mathbf{r}_1^T\mathbf{s}_j)$.

If we let $\mathbf{R}_i = \mathbf{r}_{i+1} - \mathbf{r}_1$ for $i = 1, \ldots, m-1$ be the columns of a matrix $\mathbf{R}$ and $\mathbf{S}_j = -2(\mathbf{s}_{j+1} - \mathbf{s}_1)$ for $j = 1, \ldots, n-1$ the columns of a matrix $\mathbf{S}$, the fourth group of equations can be written as $\bar{\mathbf{B}} = \mathbf{R}^T\mathbf{S}$. Assuming that the receivers and senders span 3D – that is that they do not lie for example on a line or in a plane – the matrix $\bar{\mathbf{B}}$ will have rank 3. Hence, we can express the matrix using a low rank approximation, for example, using singular value decomposition. This does, however, require that $m \geq 4$ and $n \geq 4$, since this is required for both senders and receivers to span 3D and also for $\bar{\mathbf{B}}$ to be large enough to yield this decomposition. If the decomposition gives that $\bar{\mathbf{B}} = \hat{\mathbf{R}}^T\hat{\mathbf{S}}$, we can use a full-rank 3×3 transformation matrix $\mathbf{L}$ to also express the double compaction matrix as $\bar{\mathbf{B}} = \hat{\mathbf{R}}^T\mathbf{L}^{-1}\mathbf{L}\hat{\mathbf{S}}$, with $\mathbf{R} = \mathbf{L}^{-T}\hat{\mathbf{R}}$ and $\mathbf{S} = \mathbf{L}\hat{\mathbf{S}}$. Furthermore, we choose the coordinate system such that $\mathbf{r}_1 = \mathbf{0}$ is at the origin and $\mathbf{s}_1 = \mathbf{L}\mathbf{w}$, where $\mathbf{w}$ is a 3×1 vector. For the rest of the receivers and senders we get that

$$\begin{aligned} \mathbf{r}_i &= \mathbf{L}^{-T}\hat{\mathbf{R}}_{i-1}, \quad i = 2, \ldots, m, \\ \mathbf{s}_j &= \mathbf{L}\left(\frac{\hat{\mathbf{S}}_{j-1}}{-2} + \mathbf{w}\right), \quad j = 2, \ldots, n. \end{aligned} \tag{4.8}$$

These parametrisations can now be used in the equations of the first three types. Denoting $\mathbf{H} = (\mathbf{L}^T\mathbf{L})^{-1}$ this gives

$$d_{11}^2 = (\mathbf{r}_1 - \mathbf{s}_1)^T(\mathbf{r}_1 - \mathbf{s}_1) = \mathbf{s}_1^T\mathbf{s}_1 = \mathbf{w}^T\mathbf{L}^T\mathbf{L}\mathbf{w} = \mathbf{w}^T\mathbf{H}^{-1}\mathbf{w}, \tag{4.9}$$

$$\begin{aligned} d_{1j}^2 - d_{11}^2 &= \mathbf{s}_j^T\mathbf{s}_j - \mathbf{s}_1^T\mathbf{s}_1 = \frac{1}{4}\hat{\mathbf{S}}_{j-1}^T\mathbf{L}^T\mathbf{L}\hat{\mathbf{S}}_{j-1} - \mathbf{w}^T\mathbf{L}^T\mathbf{L}\hat{\mathbf{S}}_{j-1} \\ &= \frac{1}{4}\hat{\mathbf{S}}_{j-1}^T\mathbf{H}^{-1}\hat{\mathbf{S}}_{j-1} - \mathbf{w}^T\mathbf{H}^{-1}\hat{\mathbf{S}}_{j-1}, \end{aligned} \tag{4.10}$$

$$\begin{aligned} d_{i1}^2 - d_{11}^2 &= \hat{\mathbf{R}}_{i-1}^T\mathbf{L}^{-1}\mathbf{L}^{-T}\hat{\mathbf{R}}_{i-1} - 2\mathbf{w}^T\mathbf{L}^{-1}\mathbf{L}\hat{\mathbf{R}}_{i-1} \\ &= \hat{\mathbf{R}}_{i-1}^T\mathbf{H}\hat{\mathbf{R}}_{i-1} - 2\mathbf{w}^T\hat{\mathbf{R}}_{i-1}. \end{aligned} \tag{4.11}$$

This system only contains the unknowns $\mathbf{H}$ and $\mathbf{w}$, a total of twelve unknown values, but since $\mathbf{H}$ is symmetric there are actually only nine unique unknowns. By rewriting $\mathbf{H}^{-1} = \text{adj}(\mathbf{H})/\det(\mathbf{H})$ and multiplying Equations (4.9) and (4.10) by $\det(\mathbf{H})$ all the equations can be written as polynomial equations. There are in total $m + n - 1$ equations, of which $m - 1$ are linear, $n - 1$ are of degree 3 and one of degree 4. The constraint that $m + n - 1 \geq 9$ gives the minimal cases with six receivers and four senders (or the converse) and five receivers and five senders, also denoted $(6r/4s)$ and $(5r/5s)$, respectively. Using either of these setups and the parametrisation above, the 3D position can be found

using methods from algebraic geometry. Note that in the $(5r/5s)$ case we also need that $\det(\bar{\mathbf{B}}) = 0$. This condition is used explicitly in Paper II to solve for the constant offset case (COTDOA).

4.2.2 Calibration Using TDOA and the Stratified Approach

In the case of TDOA, there are – in addition to $\mathbf{r}_i$ and $\mathbf{s}_j$ – also n unknown offsets o_j, see Equation (4.2). One approach to solve for these – as well as the anchor positions – is to use a two-tiered stratified solution method, where we first find the offsets and then solve for the anchors. If we first find all the offsets, the problem can be converted to a TOA problem and solved as explained in the previous section (if the number of senders and receivers agree). However, it is common that the first step of this approach also includes solving the following relaxed problem

$$(z_{ikj} - o_j)^2 = \mathbf{u}_i^T \mathbf{v}_j + a_i + b_j, \tag{4.12}$$

where $\mathbf{u}_i$ and $\mathbf{v}_j$ are the columns of two sought matrices $\mathbf{U} \in \mathbb{R}^{3\times m}$ and $\mathbf{V} \in \mathbb{R}^{3\times n}$, respectively, and $\mathbf{a} \in \mathbb{R}^m$ and $\mathbf{b} \in \mathbb{R}^n$ are – apart from a scalar offset – affine combinations of the columns and rows of the matrix $\mathbf{D}^{\circ 2} = \{d_{ij}^2\}$ with entries $d_{ij}^2 = (z_{ikj} - o_j)^2$. The matrices $\mathbf{U}$ and $\mathbf{V}$ are similar to $\hat{\mathbf{R}}$ and $\hat{\mathbf{S}}$ in the previous section in such way that they together form a low rank approximation of the double compaction matrix $\mathbf{M} = \mathbf{U}^T\mathbf{V}$. In this section we denote the double compaction matrix by $\mathbf{M}$, since it is described in a more general way than $\bar{\mathbf{B}}$, namely [49]

$$M_{ij} = (z_{ikj} - o_j)^2 - a_i - b_j, \tag{4.13}$$

with M_{ij} being the elements of $\mathbf{M}$. If we let $\mathbf{a}$ and $\mathbf{b}$ be the first row and column of $\mathbf{D}^{\circ 2}$, respectively, and we subtract the constant d_{11}^2 from either of them, then $\mathbf{M}$ will be the same as $\bar{\mathbf{B}}$, except for a zero row and column (and the fact that we are now using offsets as well). However, this formulation is more general in the sense that it does not depend on the first column and row of the matrix $\mathbf{D}$ to be complete. The connection between $\bar{\mathbf{B}}$ and $\mathbf{M}$ is described further in Paper III. After the computations described above, the second part of the stratified approach includes an *upgrade*, from the relaxed solution to $\mathbf{R}$, $\mathbf{S}$ and $\mathbf{o}$ (collecting all offsets o_j in $\mathbf{o}$).

To summarise the stratified approach we can say that in the first step we are looking for a set of relaxed parameters $\boldsymbol{\theta}_2 = \{\mathbf{U}, \mathbf{V}, \mathbf{b}, \mathbf{a}, \mathbf{o}\}$ and in the second step the actual sought parameter set $\boldsymbol{\theta}_1 = \{\mathbf{R}, \mathbf{S}, \mathbf{o}\}$. Different ways to solve for the first step and to upgrade are covered in Papers II–IV of this thesis.

4.2.3 RANSAC and Row and Column Extension

To solve for node positions we typically need minimal cases, for example the one mentioned above with five senders and five receivers. However, mostly we have larger problems, with more receivers and senders. These can be solved by choosing the suitable amount of rows and columns (corresponding to receivers and senders) of the measured distance matrix $\mathbf{D} = \{d_{ij}\}$, solve for the position of these, and then extend this solution to the rest of the nodes. The selection of rows and columns can be done in a RANSAC framework (see Section 2.6), in order for us to get the best possible solution.

Once the solution for the chosen sub-matrix is found, more columns and rows of the measurement matrix can be added. This has been briefly described in [7] and can also be found in the code used for [50]. Note that the measurement matrix will contain noisy measurements $\tilde{d}_{ij}$ for the TOA case and $\tilde{z}_{ij}$ for the TDOA cases. First, the solution is extended to more columns. This means that for the TOA problem the receivers $\mathbf{r}_i$ are known and the senders $\mathbf{s}_j$ are to be found; for the original TDOA problem both $\mathbf{s}_j$ and the offsets o_j are unknown; and for the relaxed problem the known values are $\mathbf{u}_i$ and a_i, while $\mathbf{v}_j$, b_j and o_j are unknown. This reduces the number of unknowns compared to the original problem, and the unknowns can be found using as many of the measurement values in that column as it requires – three for $\mathbf{s}_j$; four for $\mathbf{s}_j$ and o_j; and five for $\mathbf{v}_j$, b_j and o_j. This is done using Equation (4.1) for TOA, Equation (4.2) for the original TDOA problem and (4.12) for the relaxed TDOA problem. Clearly, only values in rows that have been solved for previously can be used, that is, rows i for which $\mathbf{r}_i$ or $\mathbf{u}_i$ and a_i have already been computed. Thereafter, the solution is tested on the remaining measurement values of that column (again, only for previously solved rows) – using the same equation – to add more inlier values. If there are more rows that are solved for than the number of values that are required to solve the equation, this as well is done in a RANSAC loop.

Once as many columns as possible are added, new rows are added in a similar fashion. The difference is that now $\mathbf{s}_j$ are known and $\mathbf{r}_i$ unknown for TOA; $\mathbf{s}_j$ and o_j are known and $\mathbf{r}_i$ unknown for the original TDOA problem; and $\mathbf{v}_j$, b_j and o_j are known and $\mathbf{u}_i$ and a_i are known for the relaxed problem. Except for the variables solved for, the procedure is the same.

Thereafter, it might be possible that even more columns can be added, as new rows have been solved for, and similarly for the rows after that. Once no more rows or columns can be added all variables that can be solved for are solved for, and the values in the measurement matrix that have not been of use are classified as outliers. Above, we started the extension with adding more rows, followed by columns, but this could as well be done in the opposite order. Figure 4.2 shows a schematic image of the procedure of row and column extension.

To give an example of how this extension can be done, we look at Equation (4.12) for the

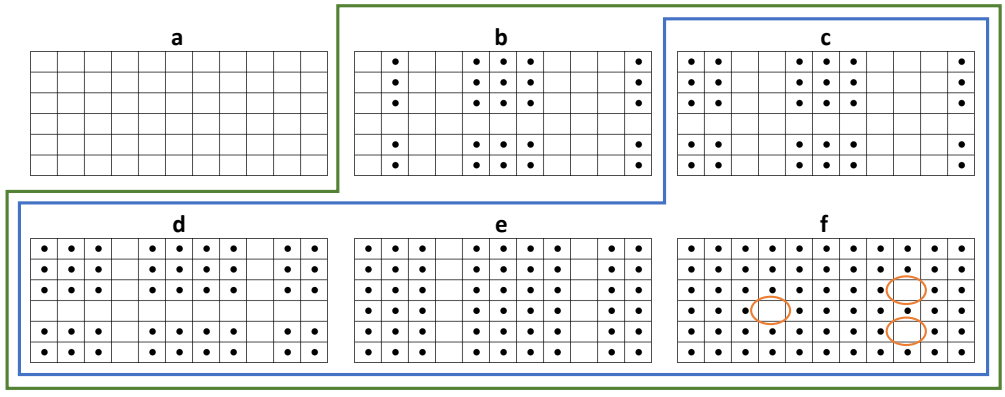


Figure 4.2: The scheme shows how to solve for node positions for larger systems. a) A measurement matrix $\{\tilde{d}_{ij}\}$ or $\{\tilde{z}_{ij}\}$ of size 6×11 is given (that is, there are six receivers and eleven senders). So far, no measurements are used and all positions are blank. Used measurements will be marked by a dot. b) A $(5r/5s)$ solver will be used so a 5×5 submatrix is chosen and these values are solved for using the minimal solver. This is done in a RANSAC fashion, indicated by the green lines. c) The solution is extended to another column using the already calculated values and for example Equation (4.1), (4.2) or (4.12). This extension is done in a RANSAC loop, indicated by the blue lines. d) The solution is extended to even more columns. e) Once there are no more columns to which the solution can be extended, the solution is extended to new rows. f) After the row extension, more columns are added, again followed by more rows etc. Once no more measurement values can be added, the process stops. Measurements that have not been used are marked by red circles. These are either outlier values or missing data points. One iteration of both the inner (blue) and the outer (green) RANSAC loop end here.

relaxed TDOA problem above. If we want to extend a partial solution to more rows, we have four unknowns – three values in $\mathbf{u}_i$ and one value in a_i. Hence, we will need values for $\mathbf{v}_j$, b_j and o_j from four (random) columns and the corresponding four measurement values z_{ikj} from the chosen row to extend the solution. For these four values of j we re-arrange the equation to

$$\begin{bmatrix} \mathbf{v}_j^T & 1 \end{bmatrix} \begin{bmatrix} \mathbf{u}_i \\ a_i \end{bmatrix} = (z_{ikj} - o_j) - b_j. \tag{4.14}$$

The unknowns are now collected in a vector and this linear system can easily be solved using the four measurements. Once this is done, the found values of $\mathbf{u}_i$ and a_i are inserted into Equation (4.12) together with the corresponding values for the j's that have not yet been used (but have been solved for previously) to count the consensus set. This is one RANSAC iteration. After this four new values of j are used and so forth. Remember that except for this, we also have an outer RANSAC loop, where the initial rows and columns are chosen.

4.2.4 Bundle Adjustment

The minimal solvers and RANSAC scheme above are good for finding an initial solution as well as an inlier set for the measurements. However, given this, the result can be improved even further using local optimisation. Once the initial solution is found, the error

function (2.76) can be formulated as

$$E(\{\tilde{d}_{ij}\},\{\mathbf{r}_i,\mathbf{s}_j\}) = \sum_{i,j}\left(\tilde{d}_{ij} - \sqrt{(\mathbf{r}_i-\mathbf{s}_j)^T(\mathbf{r}_i-\mathbf{s}_j)}\right)^2, \tag{4.15}$$

for TOA and

$$E(\{\tilde{z}_{ikj}\},\{\mathbf{r}_i,\mathbf{s}_j,o_j\}) = \sum_{i,j}\left(\tilde{z}_{ikj} - o_j - \sqrt{(\mathbf{r}_i-\mathbf{s}_j)^T(\mathbf{r}_i-\mathbf{s}_j)}\right)^2, \tag{4.16}$$

for the final step of TDOA. For the relaxed TDOA problem, we get the error function

$$E(\{\tilde{z}_{ikj}\},\{\mathbf{u}_i,\mathbf{v}_j,a_i,b_j,o_j\}) = \sum_{i,j}\left(\tilde{z}_{ikj} - o_j - \sqrt{\mathbf{u}_i^T\mathbf{v}_j + a_i + b_j}\right)^2. \tag{4.17}$$

Here, the summation indices are slightly simplified, as we rather summarise over the i,j for which $\tilde{d}_{ij}$ or $\tilde{z}_{ij}$ are inliers. Clearly, we can choose other error functions as well, see Section 2.5. Using these error functions, bundle adjustment can be performed to minimise the error and to achieve optimal estimates, see Section 2.5.2. At which stage it is best to perform the local optimisation is briefly discussed in Paper V. Remember that $\tilde{d}_{ij}$ and $\tilde{z}_{ij}$ are noisy measurements of d_{ij} and z_{ij}, respectively. Since hopefully all outliers values have been detected in the RANSAC step, we are only optimising over inliers with small noise which is assumed to be Gaussian.

4.3 Localisation for Image Data

For TOA and TDOA problems, the algorithms for mapping the environment and localising a new sensor in the scene are similar. The reason for this is that senders and receivers are represented in the same way. However, this is not the case for vision. Therefore, the localisation – or *camera resection* – and the mapping – or *triangulation* – are here divided in two sections. We start with localisation. The rest of this chapter has been inspired by [35] and [68].

When it comes to localisation, we have the following problem: the image points and the 3D points are known, while the camera matrices are unknown. The problem is solved by finding a solution to the system of equations given by

$$\lambda_j\mathbf{u}_j = \mathbf{P}\mathbf{U}_j, \qquad j = 1,\ldots,n, \tag{4.18}$$

where the camera matrix $\mathbf{P}$ is the sought value, $\mathbf{U}_j$ are the 3D points, $\mathbf{u}_j$ the corresponding image points and λ_j are unknown factors – also called depths – that arise when we make

the image points homogeneous. Again, for measured values, we are likely to have some noise in the system, such that

$$\begin{cases} \lambda_j \tilde{\mathbf{u}}_j = \mathbf{P}\mathbf{U}_j, & j = 1, ..., n, \\ \tilde{\mathbf{u}}_j = \mathbf{u}_j + \varepsilon_j, \end{cases} \tag{4.19}$$

where $\varepsilon_j \in \mathcal{N}(0, \sigma) \times \mathcal{N}(0, \sigma)$ if it is an inlier and much larger if the point is an outlier. As long as we have sufficiently many point correspondences $\mathbf{u}_j$ and $\mathbf{U}_j$ we can solve for the unknowns using a method called *direct linear transform* (DLT). Looking at Equation (4.18), each 2D-3D correspondence does in fact give $3n$ equations, one for each coordinate. Furthermore, there are eleven unknowns in $\mathbf{P}$ (given that the scale is arbitrary) and one extra unknown λ_j for each 2D-3D point pair. Hence, we need

$$3n \geq 11 + n \quad \Leftrightarrow \quad n \geq 5.5, \tag{4.20}$$

that is at least $n = 6$ points to solve for the camera matrices, as well as the depths. Now, if we denote the ith row of $\mathbf{P}$ by $\mathbf{p}_i^T$, the three equations coming from Equation (4.18) will be

$$\begin{cases} \mathbf{U}_j^T \mathbf{p}_1 - \lambda_j u_{j1} & = 0, \\ \mathbf{U}_j^T \mathbf{p}_2 - \lambda_j u_{j2} & = 0, \\ \mathbf{U}_j^T \mathbf{p}_3 - \lambda_j \cdot 1 & = 0, \end{cases} \tag{4.21}$$

where u_{jk} represents the kth coordinate of $\mathbf{u}_j$. These equations can be written using matrices, and for six different values of j we get a 18×18 matrix with known values and a 18×1 vector of unknowns

$$\begin{bmatrix} \mathbf{U}_1^T & \mathbf{0} & \mathbf{0} & -u_{11} & 0 & 0 & \dots & 0 \\ \mathbf{0} & \mathbf{U}_1^T & \mathbf{0} & -u_{12} & 0 & 0 & \dots & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_1^T & 1 & 0 & 0 & \dots & 0 \\ \mathbf{U}_2^T & \mathbf{0} & \mathbf{0} & 0 & -u_{21} & 0 & \dots & 0 \\ \mathbf{0} & \mathbf{U}_2^T & \mathbf{0} & 0 & -u_{22} & 0 & \dots & 0 \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_2^T & 0 & 1 & 0 & \dots & 0 \\ & \vdots & & \vdots & & & \ddots & \\ \mathbf{U}_6^T & \mathbf{0} & \mathbf{0} & 0 & 0 & 0 & \dots & -u_{61} \\ \mathbf{0} & \mathbf{U}_6^T & \mathbf{0} & 0 & 0 & 0 & \dots & -u_{62} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_6^T & 0 & 0 & 0 & \dots & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \\ \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_6 \end{bmatrix} = \mathbf{M}\mathbf{v} = \mathbf{0}. \tag{4.22}$$

Here, $\mathbf{0}$ denotes a vector of zeroes (to the left 1×4 vectors, to the right a 18×1 vector). As the positions of $\mathbf{U}_j$ and $\mathbf{u}_j$ might contain noise, it is unlikely that we can find an exact solution to Equation (4.22). Instead, we solve for

$$\min_{\|\mathbf{v}\|^2 = 1} \|\mathbf{M}\mathbf{v}\|^2 = \min_{\mathbf{v}^T\mathbf{v} = 1} \mathbf{v}^T \mathbf{M}^T \mathbf{M} \mathbf{v}, \tag{4.23}$$

where the constraint $\|\mathbf{v}\|^2 = 1$ is added to get a unique solution although the scale is arbitrary. Actually, there will still be an ambiguity concerning the sign of $\mathbf{v}$, since $\|\mathbf{v}\|^2 = \|-\mathbf{v}\|^2$, but then we choose the sign that gives a positive solution for the depths λ_j. For a solution, we need the gradients of $\mathbf{v}^T\mathbf{v} - 1$ and $\mathbf{v}^T\mathbf{M}^T\mathbf{M}\mathbf{v}$ to be parallel, that is [10]

$$\nabla(\mathbf{v}^T\mathbf{M}^T\mathbf{M}\mathbf{v}) = \mu\nabla(\mathbf{v}^T\mathbf{v} - 1) \quad \Rightarrow \quad \mathbf{M}^T\mathbf{M}\mathbf{v} = \mu\mathbf{v}, \tag{4.24}$$

for some value μ. Now, we can identify an eigenvalue problem, where the solution $\mathbf{v}$ is an eigenvector to $\mathbf{M}^T\mathbf{M}$. However, this can be found through SVD of $\mathbf{M}$, with $\mathbf{M} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^T$, since

$$\mathbf{M}^T\mathbf{M} = \mathbf{V}\mathbf{\Sigma}^T\mathbf{W}^T\mathbf{W}\mathbf{\Sigma}\mathbf{V}^T = \mathbf{V}\mathbf{\Sigma}^T\mathbf{\Sigma}\mathbf{V}^T. \tag{4.25}$$

Comparing this to Equation (2.29), we see that $\mathbf{\Lambda} = \mathbf{\Sigma}^T\mathbf{\Sigma}$ contains the eigenvalues of $\mathbf{M}^T\mathbf{M}$ and $\mathbf{V}$ contains the eigenvectors. Furthermore, in order to minimise Equation (4.23), we should choose the eigenvector corresponding to the smallest eigenvalue, since

$$\|\mathbf{M}\mathbf{v}\| = \|\mu\mathbf{v}\| = |\mu|\|\mathbf{v}\| = |\mu|. \tag{4.26}$$

This means that the solution is given by the last column of the matrix $\mathbf{V}$ and from the first twelve values of that vector we can find $\mathbf{P}$. Note that the scale is still arbitrary.

The problem above can be ill-conditioned if the values of $\mathbf{U}_j$ and $\mathbf{u}_j$ are large. This can be solved by normalising the points before forming the matrix $\mathbf{M}$. It is also worth noting that $\mathbf{M}$ can be created using more than six point correspondences, if more are known, but not less. [35]

4.3.1 Finding the Intrinsic and Extrinsic Parameters

Once the camera matrix $\mathbf{P}$ is obtained, we can factorise it in order to find $\mathbf{K}$, $\mathbf{R}$ and $\mathbf{t}$. From Equation (3.29) we know that $\mathbf{P} = \begin{bmatrix}\mathbf{KR} & \mathbf{Kt}\end{bmatrix}$. Focusing on the left 3×3 matrix $\mathbf{KR}$, this can be factorised into an upper triangular matrix $\mathbf{K}$ and an orthogonal matrix $\mathbf{R}$ using RQ decomposition, see Section 2.3.1. Once $\mathbf{K}$ is known, we can obtain $\mathbf{t}$ from back-substitution using the rightmost column $\mathbf{Kt}$. For more details, see [35].

4.4 Triangulation for Image Data

In the previous section we showed how the position of the camera can be found if we know a number of point correspondences. In that case, the image points and the 3D points were known, while camera matrices were unknown. When it comes to mapping, we have the opposite problem: the image points and the cameras are known, while the 3D points are

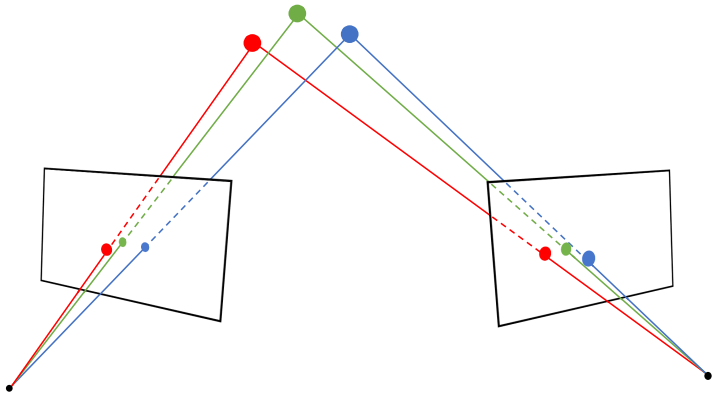


Figure 4.3: Triangulation of three points given two images. The corresponding image points $\mathbf{u}_i$ as well as the cameras $\mathbf{P}_i$ are given. Each 3D point $\mathbf{U}$ is obtained by intersecting the lines from each camera centre through the corresponding image point.

unknown. This problem is called *triangulation* and refers – just as the name suggests – to deciding distances using triangles. A small triangulation example is shown in Figure 4.3.

Assuming that we have corresponding image points $\mathbf{u}_i$ taken by the cameras $\mathbf{P}_i$, we for each corresponding feature point pair i have,

$$\lambda_i \mathbf{u}_i = \mathbf{P}_i \mathbf{U} \tag{4.27}$$

(or for noisy values $\lambda_i \tilde{\mathbf{u}}_i = \mathbf{P}_i \mathbf{U}$) where $\mathbf{U}$ is the unknown 3D feature point that the image points $\mathbf{u}_i$ depict. If the camera matrices $\mathbf{P}_i$ are known, one can draw a line from each camera centre, through the corresponding image point, out in space. The lines will lie in a plane and intersect in a point – the point $\mathbf{U}$ [35]. To find the coordinates of $\mathbf{U}$, we need a total of n image point correspondences. This will give $3n$ equations and $n + 3$ unknowns, which means that we require

$$3n \geq n + 3 \quad \Leftrightarrow \quad n \geq 1.5, \tag{4.28}$$

that is, it is enough with two images points to solve for $\mathbf{U}$.

If we collect the equations for the different coordinates and points in a matrix, as for localisation, we get the equation

$$\begin{bmatrix} \mathbf{P}_1 & -\mathbf{u}_1 & \mathbf{0} \\ \mathbf{P}_2 & \mathbf{0} & -\mathbf{u}_2 \end{bmatrix} \begin{bmatrix} \mathbf{U} \\ \lambda_1 \\ \lambda_2 \end{bmatrix} = \mathbf{M}\mathbf{v} = \mathbf{0}. \tag{4.29}$$

Again, in reality, there is in most cases some noise in the measurements and the 3D points will have to be estimated in a least squares sense. This can be done using DLT, as for the

case of localisation in the previous section. Just as before, more point correspondences can be used, but it will give an over-determined system. How the point correspondences can be found will be discussed later, in Section 5.1.

4.5 Structure from Motion and Simultaneous Localisation and Mapping

We have previously in this chapter explained how mapping and localisation can be done separately. For sound and radio data localisation corresponds to finding a sender position $\mathbf{s}_i$ and mapping is done by finding the receivers $\mathbf{r}_j$. The correspondence in vision is finding the cameras $\mathbf{P}_i$ for localisation and the 3D points $\mathbf{U}_j$ for mapping. When both mapping and localisation are performed at once, this is referred to as SfM or visual SLAM for vision. For sound and radio, a solution to this problem – the structure from sound or automatic calibration problem – was given in Section 4.1. Hence, this section only covers the vision case. It is worth mentioning, though, that the principles often are the same for different types of sensor data.

4.5.1 Epipolar Geometry and SfM for Two Uncalibrated Cameras

In many cases when we want to use vision to map an environment we need to use SfM algorithms, since it is uncommon that the camera positions are known. When doing the triangulation and the camera resection separately, the systems can be solved linearly. However, now both the cameras and 3D points are unknown and we get the equations

$$\lambda_{ij}\mathbf{u}_{ij} = \mathbf{P}_i\mathbf{U}_j, \quad \text{or} \quad \lambda_{ij}\tilde{\mathbf{u}}_{ij} = \mathbf{P}_i\mathbf{U}_j, \tag{4.30}$$

where $\mathbf{u}_{ij}$ is the image point retrieved from projecting $\mathbf{U}_j$ in camera $\mathbf{P}_i$ and $\tilde{\mathbf{u}}_{ij}$ is the corresponding point with noise. If the cameras are uncalibrated, they (and the 3D points) can only be determined up to a projective transformation, since

$$\lambda_{ij}\mathbf{u}_{ij} = \mathbf{P}_i\mathbf{U}_j \qquad \Leftrightarrow \qquad \lambda_{ij}\mathbf{u}_{ij} = \mathbf{P}_i\mathbf{H}\mathbf{H}^{-1}\mathbf{U}_j = \hat{\mathbf{P}}_i\hat{\mathbf{U}}_i, \tag{4.31}$$

where $\mathbf{H}$ is a projective transformation, $\hat{\mathbf{P}}_i = \mathbf{P}_i\mathbf{H}$ and $\hat{\mathbf{U}}_j = \mathbf{H}^{-1}\mathbf{U}_j$. If the cameras are calibrated, however, they are determined up to a similarity transformation – representing a change of rotation, translation and scale. These seven degrees of freedom are referred to as *gauge freedom*.

Independently of whether the cameras are calibrated or not, it turns out that we can eliminate the scene points in Equation (4.31) and express the solution of the cameras using

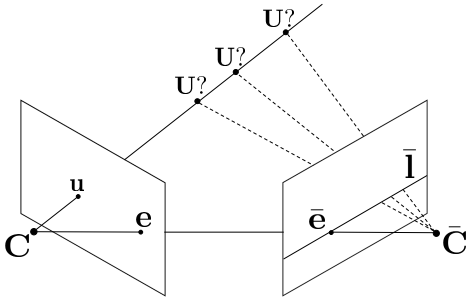


Figure 4.4: The figure shows two images and the camera positions $\mathbf{C}$ and $\bar{\mathbf{C}}$ that they were taken from. The image point $\mathbf{u}$ depicts the 3D point $\mathbf{U}$ that is located somewhere along the viewing ray $\mathbf{U}(s)$. The image of the viewing ray in the second camera is the epipolar line $\bar{\mathbf{l}}$ and the images of the camera centres in the other camera are, respectively, the epipoles $\mathbf{e}$ and $\bar{\mathbf{e}}$.

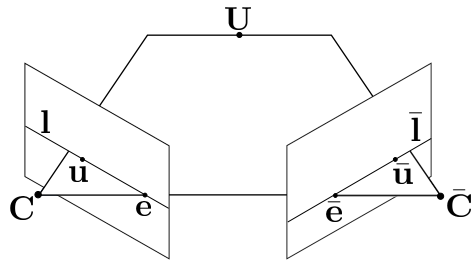


Figure 4.5: The figure shows how the camera centres $\mathbf{C}$ and $\bar{\mathbf{C}}$ and the 3D point $\mathbf{U}$ all lie on a plane – the epipolar plane. The intersections of this plane with the image planes give the two epipolar lines $\mathbf{l}$ and $\bar{\mathbf{l}}$ and both the image points $\mathbf{u}$ and $\bar{\mathbf{u}}$ and the epipoles $\mathbf{e}$ and $\bar{\mathbf{e}}$ lie on these lines, respectively. The epipoles also lie on the baseline, which is the line joining the two camera centres.

only the image points. For this, we will restrict ourselves to two cameras $\mathbf{P}$ and $\bar{\mathbf{P}}$. The geometric constraints that will arise are referred to as *epipolar geometry*.

Since the cameras only can be determined up to a projective transformation and we can choose the coordinate system we can always assume that the cameras are of the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}, \qquad \bar{\mathbf{P}} = \begin{bmatrix} \mathbf{A} & \mathbf{t} \end{bmatrix}, \tag{4.32}$$

where $\mathbf{A}$ is a general 3×3 matrix. A point $\mathbf{u}$ in the first image is the projection of a 3D point that lies somewhere along the line going through the camera centre of $\mathbf{P}$ – which we will call $\mathbf{C}$ and which in this case is the origin – and the first image point $\mathbf{u}$. This line (also called the *viewing ray*) can be described as

$$\mathbf{U}(s) = \begin{bmatrix} \mathbf{u} \\ s \end{bmatrix}. \tag{4.33}$$

The projection of this line into the second camera will be a line in that image plane, given by

$$\bar{\mathbf{P}}\mathbf{U}(s) = \begin{bmatrix} \mathbf{A} & \mathbf{t} \end{bmatrix} \begin{bmatrix} \mathbf{u} \\ s \end{bmatrix} = \mathbf{A}\mathbf{u} + s\mathbf{t}. \tag{4.34}$$

This line is called the *epipolar line* of $\mathbf{u}$.

Different image points $\mathbf{u}$ will give different epipolar lines in the second image. However, since all the viewing rays $\mathbf{U}(s)$ go through the camera centre of the first camera $\mathbf{C}$, the projection of this point in the second camera will be included in all epipolar lines. This point $\bar{\mathbf{e}} = \bar{\mathbf{P}}\mathbf{C}$ is called the *epipole*. Similarly, the projection of the camera centre for

the second camera $\bar{\mathbf{C}}$ in the first image is also an epipole $\mathbf{e} = \mathbf{P}\bar{\mathbf{C}}$. Figure 4.4 shows an illustration of the connection between the epipoles and the camera centres as well as the viewing ray and the epipolar line. Furthermore, both $\mathbf{e}$ and $\bar{\mathbf{e}}$ will lie on the line between $\mathbf{C}$ and $\bar{\mathbf{C}}$ – called the *baseline* – and the plane that contains this line and the 3D point $\mathbf{U}$ is called the *epipolar plane*. The intersections of the epipolar plane with the image planes give the epipolar lines (the epipolar line in the first image is defined analogously with the one in the second image). This plane is illustrated in Figure 4.5.

The epipolar line $\mathbf{A}\mathbf{u} + s\,\mathbf{t}$ can also be expressed in $\mathcal{P}^2$ as $\bar{\mathbf{l}} = \mathbf{t} \times (\mathbf{A}\mathbf{u}) = [\mathbf{t}]_\times \mathbf{A}\mathbf{u}$, with $[\mathbf{t}]_\times$ being the matrix representing the cross product with $\mathbf{t}$. Equivalently, it can be described using the epipole in the second image, $\bar{\mathbf{l}} = [\bar{\mathbf{e}}]_\times \mathbf{A}\mathbf{u}$ [35]. Since any point $\bar{\mathbf{u}}$ in the second image that corresponds to the same 3D point $\mathbf{U}$ as $\mathbf{u}$ lies along this line, such a point much fulfil

$$\bar{\mathbf{u}}^T \bar{\mathbf{l}} = \bar{\mathbf{u}}^T [\bar{\mathbf{e}}]_\times \mathbf{A}\mathbf{u} = \bar{\mathbf{u}}^T \mathbf{F}\mathbf{u} = 0, \tag{4.35}$$

where we denote the matrix $[\bar{\mathbf{e}}]_\times \mathbf{A}$ by $\mathbf{F}$. This is called the *fundamental matrix* and encodes the relationship between corresponding points in the two images without including the 3D points. We note that $\mathbf{F}$ has rank 2 such that $\det(\mathbf{F}) = 0$, since $[\bar{\mathbf{e}}]_\times$ has rank 2.

The fundamental matrix contains nine elements, but since the scale is arbitrary and $\det(\mathbf{F}) = 0$, it has seven degrees of freedom. If we have image point pairs from the two images, where $\mathbf{x}_i$ in the first image matches $\bar{\mathbf{x}}_i$ in the second image, each image point pair $(\mathbf{x}_i, \bar{\mathbf{x}}_i)$ gives rise to one equation

$$\begin{aligned} 0 = \bar{\mathbf{x}}_i^T \mathbf{F} \mathbf{x}_i = \quad & F_{11}\bar{x_{i1}}x_{i1} + F_{12}\bar{x_{i1}}x_{i2} + F_{13}\bar{x_{i1}}x_{i3} \\ & + F_{21}\bar{x_{i2}}x_{i1} + F_{22}\bar{x_{i2}}x_{i2} + F_{23}\bar{x_{i2}}x_{i3} \\ & + F_{31}\bar{x_{i3}}x_{i1} + F_{32}\bar{x_{i3}}x_{i2} + F_{33}\bar{x_{i3}}x_{i3}, \end{aligned} \tag{4.36}$$

where F_{ij} are the elements of $\mathbf{F}$ and x_{ij} is the jth value if $\mathbf{x}_i$. To be able to solve for $\mathbf{F}$ linearly we ignore the condition on the determinant; hence, we need in total eight equations and correspondences. Then, we can set up a linear matrix equation, similar to the ones we used for triangulation and camera resection. Collecting all unknowns F_{ij} in a vector $\mathbf{v}$ and the known values in a matrix $\mathbf{M}$ gives

$$\begin{bmatrix} \bar{x_{11}}x_{11} & \bar{x_{11}}x_{12} & \bar{x_{11}}x_{13} & \bar{x_{12}}x_{11} & \cdots & \bar{x_{13}}x_{13} \\ \bar{x_{21}}x_{21} & \bar{x_{21}}x_{22} & \bar{x_{21}}x_{23} & \bar{x_{22}}x_{21} & \cdots & \bar{x_{23}}x_{23} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \bar{x_{81}}x_{81} & \bar{x_{81}}x_{82} & \bar{x_{81}}x_{83} & \bar{x_{82}}x_{81} & \cdots & \bar{x_{83}}x_{83} \end{bmatrix} \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ \vdots \\ F_{33} \end{bmatrix} = \mathbf{M}\mathbf{v} = \mathbf{0}. \tag{4.37}$$

As for Equation (4.22) we solve this using DLT. This method for computing the fundamental matrix is sometimes referred to as the *eight-point algorithm* (although – again – more

points can be used if desired). However, due to noise, it is not certain that the resulting matrix will have a zero determinant. If we call the matrix that is estimated by DLT $\tilde{\mathbf{F}}$, we can ensure that $\mathbf{F}$ has rank 2, by letting

$$\mathbf{F} = \mathbf{W} \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^T, \tag{4.38}$$

if $\tilde{\mathbf{F}} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}$ is an SVD of $\tilde{\mathbf{F}}$ and σ_1 and σ_2 are the two largest singular values in $\mathbf{\Sigma}$.

Once the fundamental matrix is computed, the camera matrix $\bar{\mathbf{P}}$ can be obtained from it. The cameras can only be determined up to a projective transformation, but they may be chosen as [35]

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}, \qquad \bar{\mathbf{P}} = \begin{bmatrix} [\bar{\mathbf{e}}]_\times \mathbf{F} & \bar{\mathbf{e}} \end{bmatrix}, \tag{4.39}$$

and a general formula is given by

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}, \qquad \bar{\mathbf{P}} = \begin{bmatrix} [\bar{\mathbf{e}}]_\times \mathbf{F} + \bar{\mathbf{e}}\mathbf{v}^T & \mu\bar{\mathbf{e}} \end{bmatrix}, \tag{4.40}$$

where $\mathbf{v}$ is any 3×1 vector and μ is a non-zero scalar. Since the epipole $\bar{\mathbf{e}}$ lies on the epipolar line we know that $\mathbf{F}^T\bar{\mathbf{e}} = 0$ and this means that $\bar{\mathbf{e}}$ can be found by computing the null space of $\mathbf{F}^T$. From this, we have retained all of $\bar{\mathbf{P}}$ (up to a projective transformation).

The structure from motion problem is not yet solved, though. First, we have to find the position of the 3D point $\mathbf{U}$. However, since the cameras now are known, this can be done using triangulation as described in Section 4.4.

4.5.2 SfM for Two Calibrated Cameras

In the previous section we derived the fundamental matrix for two general, uncalibrated cameras. If the calibration matrices $\mathbf{K}$ and $\bar{\mathbf{K}}$ are known, the calibrated cameras will be

$$\mathbf{P} = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix}, \qquad \bar{\mathbf{P}} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}, \tag{4.41}$$

and the corresponding normalised image points $\mathbf{u}' = \mathbf{K}^{-1}\mathbf{u}$ and $\bar{\mathbf{u}}' = \bar{\mathbf{K}}^{-1}\bar{\mathbf{u}}$. The correspondence to the fundamental matrix for calibrated cameras is called the *essential matrix* $\mathbf{E}$, and the correspondence to Equation (4.35) is

$$(\bar{\mathbf{u}}')^T\mathbf{E}\mathbf{u}' = 0. \tag{4.42}$$

Substituting for the uncalibrated points we see that $\bar{\mathbf{u}}^T\bar{\mathbf{K}}^{-T}\mathbf{E}\mathbf{K}^{-1}\mathbf{u} = 0$ and so the relationship between the fundamental and the essential matrix is $\mathbf{E} = \bar{\mathbf{K}}^T\mathbf{F}\mathbf{K}$. Furthermore, the essential matrix has the form [35]

$$\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}. \tag{4.43}$$

Since the matrix only consists of a translation vector and a rotation matrix, and the scale is arbitrary, $\mathbf{E}$ has five degrees of freedom. Except for the constraint on the determinant – similar to $\mathbf{F}$ – that $\det(\mathbf{E}) = 0$, the two singular values of $\mathbf{E}$ has to be equal. Ignoring the non-linear constraints we can solve for $\mathbf{E}$ using eight point correspondences, just as we did with $\mathbf{F}$. However, we will now let

$$\mathbf{E} = \mathbf{W} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{V}^T, \tag{4.44}$$

where we get the singular value decomposition $\tilde{\mathbf{E}} = \mathbf{W}\mathbf{\Sigma}\mathbf{V}^T$ from the by DLT estimated essential matrix $\tilde{\mathbf{E}}$.

Once $\mathbf{E}$ is obtained, we want to compute the camera matrix $\bar{\mathbf{P}}$. This can be done by factorising the essential matrix into $\mathbf{E} = \mathbf{SR}$, where $\mathbf{S}$ is a skew symmetric matrix and $\mathbf{R}$ is a rotation. When finding the camera matrix from the fundamental matrix we had a projective ambiguity. For the essential matrix we will instead have four different camera matrices that fulfills the requirements. For the computation of $\bar{\mathbf{P}}$ the following matrix will be used

$$\mathbf{\Phi} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.45}$$

Given this, and the SVD in Equation (4.44), the four possible solutions for the camera are [35]

$$\begin{aligned} \bar{\mathbf{P}}^{(1)} &= \begin{bmatrix} \mathbf{W}\mathbf{\Phi}\mathbf{V}^T & \mathbf{w}_3 \end{bmatrix}, \quad \bar{\mathbf{P}}^{(2)} = \begin{bmatrix} \mathbf{W}\mathbf{\Phi}\mathbf{V}^T & -\mathbf{w}_3 \end{bmatrix}, \\ \bar{\mathbf{P}}^{(3)} &= \begin{bmatrix} \mathbf{W}\mathbf{\Phi}^T\mathbf{V}^T & \mathbf{w}_3 \end{bmatrix}, \quad \bar{\mathbf{P}}^{(4)} = \begin{bmatrix} \mathbf{W}\mathbf{\Phi}^T\mathbf{V}^T & -\mathbf{w}_3 \end{bmatrix}, \end{aligned} \tag{4.46}$$

where $\mathbf{w}_3$ is the last column of $\mathbf{W}$. Despite the ambiguity, only one of these four cameras represents a solution where the 3D points are in front of both cameras. Hence, we can find the correct camera by computing the 3D points $\mathbf{U}_j$ (in the same way as earlier) for all cameras $\bar{\mathbf{P}}^{(i)}$ and then select the camera for which all points are in front of both cameras, that is, points such that the depth λ_{ij} is positive.

The Five Point Algorithm

Since the essential matrix only has five degrees of freedom, it can be solved for using fewer point correspondences. However, this requires that we include the non-linear constraints. In this section we briefly explain a way to solve for the essential matrix called the *five point algorithm*. For further details, see [65].

Except for Equation (4.42) and the requirement that $\det(\mathbf{E}) = 0$, it also holds that for any essential matrix $\mathbf{E}$ [24]

$$\mathbf{E}\mathbf{E}^T\mathbf{E} - \frac{1}{2}\text{trace}(\mathbf{E}\mathbf{E}^T)\mathbf{E} = 0. \tag{4.47}$$

To solve for $\mathbf{E}$ using only five point correspondences, we can set up a system $\mathbf{M}\mathbf{v} = \mathbf{0}$, as before, but this time with only five points. This will not be enough to find the values in $\mathbf{E}$, but the matrix $\mathbf{E}$ can be expressed using the four dimensional nullspace of $\mathbf{M}$, such that

$$\mathbf{E} = \alpha_1\mathbf{E}_1 + \alpha_2\mathbf{E}_2 + \alpha_3\mathbf{E}_3 + \alpha_4\mathbf{E}_4, \tag{4.48}$$

where $\mathbf{E}_i$ is the ith basis vector for the nullspace, reshaped into a matrix. Now, there are only the four unknowns α_i and since the scale is arbitrary we can set $\alpha_4 = 1$, leaving us with three unknowns.

By using the expression for $\mathbf{E}$ from Equation (4.48) in Equation (4.47) and $\det(\mathbf{E}) = 0$ we get a system of ten polynomial equations. By factorising these into a coefficient matrix $\mathbf{M}$ and a monomial vector $\mathbf{b}$, we can then solve for α_i using, for example, the action matrix method described in Section 2.4.2.

4.5.3 SfM for Several Points and Cameras

In the previous section we saw how both two cameras and a number of 3D points could be found using only two images. To create a 3D reconstruction of a scene, however, requires much more images. There are several different ways to solve the SfM problem for larger scenes and we will only briefly touch upon a few of them.

One way to solve for more cameras and points is to start out just as we did in the previous section, by solving for two cameras and their corresponding points. Then, one can take an iterative approach, where a new image and camera is added using the overlap with the already reconstructed 3D points using camera resection, according to Section 4.3. Thereafter new points that are common between that last image (or camera) and one of the previous images are added using triangulation, described in Section 4.4. After that is done, a new image is added, and so forth.

The pipeline described above is called *sequential structure from motion*, as the images are added one by one. It is common that error accumulates for this system, such that a 3D point that is reconstructed towards the end, might not end up where it would have been if it was reconstructed early in the process. This is known as the *loop closure problem*. There are several ways to solve for loop closure, but this will not be covered here (though, it can be seen as a matching and merging problem, see Chapter 5). It is, however, common that

an additional step is added to the pipeline, where bundle adjustment is performed over all points and cameras, to decrease the total error, see Section 2.5.2. The error function for this would be

$$E(\tilde{\mathbf{u}}_{ij}, \{\mathbf{P}_i, \mathbf{U}_j\}) = \sum_{ij} \left\| \tilde{\mathbf{u}}_{ij} - \frac{\mathbf{P}_i \mathbf{U}_j}{\lambda_{ij}} \right\|^2. \tag{4.49}$$

Usually, the term $\|\tilde{\mathbf{u}}_{ij} - \mathbf{P}_i\mathbf{U}_j/\lambda_{ij}\|$ is referred to as the *reprojection error*. This is illustrated in Figure 1.1.

Some examples of sequential SfM pipelines that are commonly used are COLMAP [79], Bundler [85, 86] and VisualSFM [98–100]. There are also examples of *non-sequential structure from motion*, where the camera orientations for all images are computed first, after which the 3D points are estimated [28]. This approach is less sensitive to loop closure problems, as the rotational consistency for all images are taken into account simultaneously, dividing the errors more evenly.

The SfM problem and its solutions have been developed in the computer vision community to estimate the structure of the scene and the motion of the camera using only images. Simultaneously, similar problems have been studied in other fields. The previously mentioned SLAM pipeline for estimating map parameters and sensor motion originates from the robotics community. Here, several type of sensors can be used, for example ultrasound, laser and images. In the latter case, it is called visual SLAM. Some examples of commonly used visual SLAM systems are PTAM [41], DTAM [64], ORB-SLAM [63], ORB-SLAM2 [62] and ORB-SLAM3 [16]. As the term SLAM can be used for several different types of sensors, the automatic calibration using sound in Section 4.1 is an example on SLAM.

Chapter 5

Map Merging

The previous chapters have described mapping, localisation and some methods needed for that. In this chapter we will discuss one matter that can improve the localisation in a known environment. Except for a good algorithm and exact measurements, the localisation is highly dependent on the description of the environment, that is, the map. A map consists of a number of feature points and each feature point has a descriptor and a position. In the case of TOA or TDOA each point would be a receiver, with an ID as the descriptor and the position in 3D as the position. For image data, the position would be a 3D point as well, while the descriptor could be a vector describing the area or volume around that point (in an image or in 3D). The map would be the set of all such feature points.

The more exact our map is, the better will the localisation be. Furthermore, the more measurements we have when creating our map, the better will it be. Therefore, the knowledge of how to merge maps is important. If we can merge maps, we can use this to add information – represented as a new map – when we get hold of new information about the environment.

Map merging has previously been used in, for example, the fields of loop closure [97] and collaborative SLAM [111]. In collaborative SLAM, several cameras are used simultaneously and the problem can be simplified by initialising the cameras such that their relative positions are known. This makes the problem similar to that of loop closure, where the beginning and end of an SfM map should be merged. In the merging papers of this thesis, we have no common initialisation and the mappings are assumed to be done at different times. An image illustrating the idea of map merging can be seen in Figure 5.1. Now, we will discuss some subjects that are of importance when it comes to map merging, namely feature extraction, feature matching and registration.

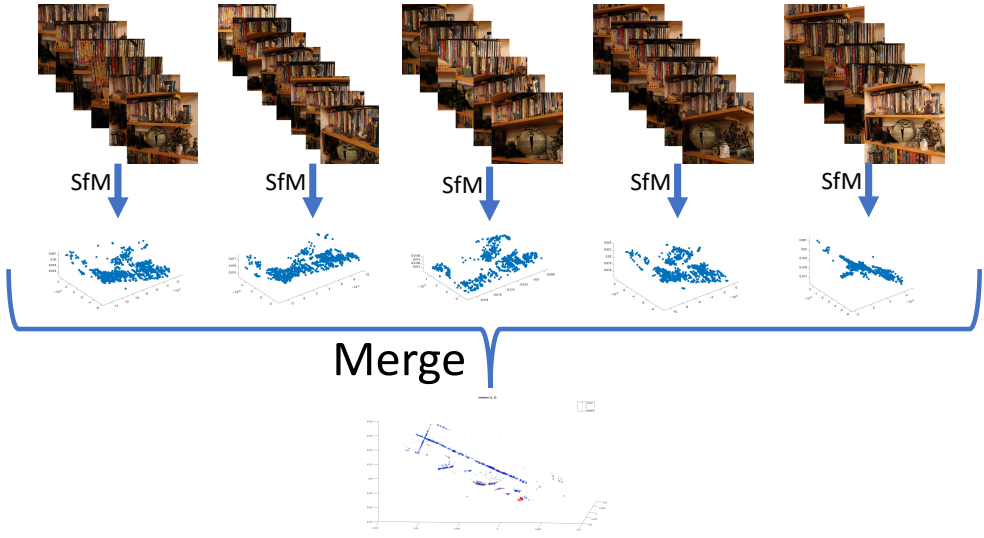


Figure 5.1: An overview of the idea of map merging and how it can be used to improve map quality. First, individual map estimates consisting of point clouds with descriptors are created from several images using SfM. Then, these maps can be merged in order to obtain a global map with lower variance, as the one in the bottom of the image.

5.1 Feature Extraction and Matching

In Sections 4.4 and 4.5, we assumed that point correspondences in images were given. However, this is usually not the case, and in order to triangulate points in 3D, point matches have to be found. This is also, for example, needed for alignment of images and for finding stereo correspondences. The correspondences are usually found by first identifying *feature points* or *landmarks* separately in the different images. Such landmarks are points that are interesting and can be distinguished from other points, for example edges and corners – points where the gradient in the image is large. An early detector of corners in images is the Harris corner detector [34]. In general, the points need to be detected as well as described, in order to be compared to other points. An example of good and less good points to match are given in Figure 5.2.

Some desired properties of feature points are that they should be invariant to scale, rotation and be recognisable from different viewpoints. Furthermore, they need to be described in some way, using some *feature descriptor* that can be compared between images. There are several known methods to find and describe such feature points. In this thesis, mainly ORB [75] has been used. Other common feature descriptors are SIFT [60], BRIEF [15], SURF [8] and there are several more. The descriptors look slightly different, but mostly they consist of a vector of numbers of some type which describe the pixel and its surroundings. It is also common that the descriptors consider several different scale space representations – in some way – to be scale invariant, and that the rotation invariance is accounted for. All

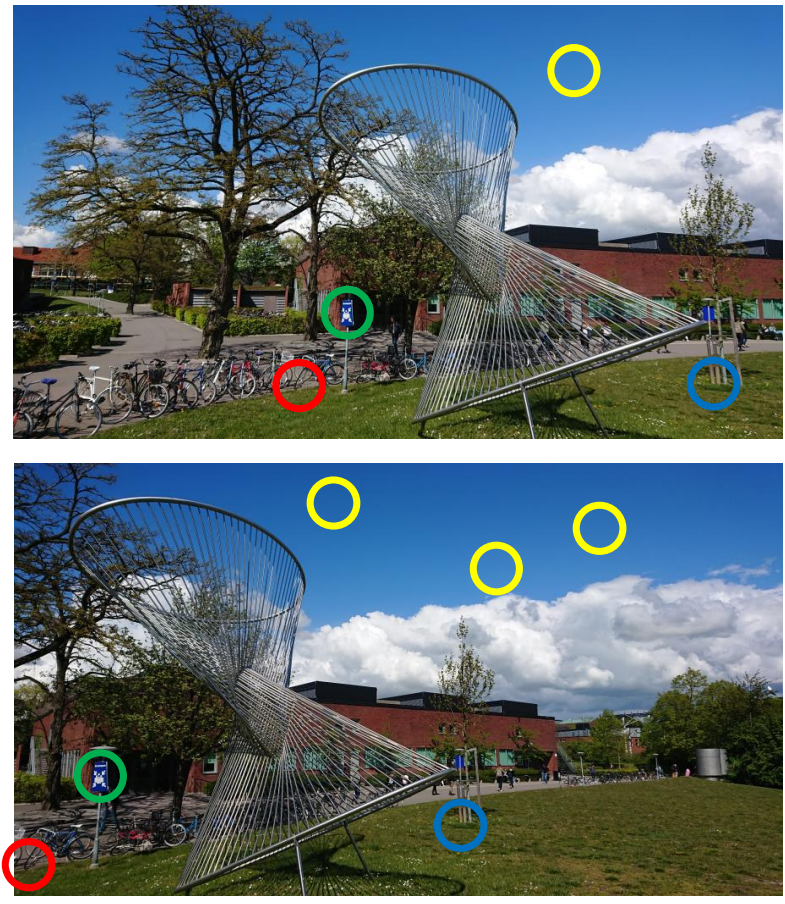


Figure 5.2: The figure shows two images and some matching points. The green, red and blue circles mark points that can be good for matching, as they are unique. The yellow circles show points that are less good, as they are not unique.

the descriptors that are mentioned so far are hand-crafted.

Over the last years many new descriptors have been developed using deep learning. The FAST descriptor [74] has partly been developed with machine learning techniques. Examples of descriptors using deep learning are DeepDesc [83], LIFT [103] and LF-Net [69]. These solve the problem of finding a good descriptor but work on image patches, like, for example, SIFT does [90]. However, there are also examples of methods that work on the whole image and solve both detection and description of features, such as SuperPoint [25]. For more examples of both hand-crafted and learned feature descriptors, see [90].

Once the feature extraction is done it remains to match these features, for example between two images. Independently of which feature is used, two feature descriptors can be compared using the Euclidean distance. One straightforward way to establish tentative matches

would be to compare each feature in the first image to those of the second image and say that it is a match if the distance between their descriptors is below a certain threshold. If we call the two images $\mathbf{X}$ and $\mathbf{Y}$ and the found features $(\mathbf{x}_i, \mathbf{f}_i)_{i=1}^{N}$ and $(\mathbf{y}_j, \mathbf{g}_j)_{j=1}^{M}$, respectively, ($\mathbf{x}_i$ and $\mathbf{y}_j$ denoting the positions and $\mathbf{f}_i$ and $\mathbf{g}_j$ the descriptors), this would mean that $\mathbf{x}_k$ and $\mathbf{y}_l$ match if

$$\|\mathbf{f}_k - \mathbf{g}_l\|_2 < t_1, \tag{5.1}$$

where the t_1 is the threshold. The distance could also be measured by some other metric, for example the Hamming distance for binary descriptors.

This method could, however, lead to many false matches if the threshold is too high and the opposite if it is too low. Also, each point in $\mathbf{X}$ could match to several points in $\mathbf{Y}$. Another option would therefore be to say that $(\mathbf{x}_k, \mathbf{f}_k)$ matches to the nearest neighbour in $\mathbf{Y}$, that is, $(\mathbf{y}_l, \mathbf{g}_l)$ is a match if

$$\text{argmin}_j \|\mathbf{f}_k - \mathbf{g}_j\| = l. \tag{5.2}$$

Since every point in $\mathbf{X}$ has a nearest neighbour in $\mathbf{Y}$, while there in reality are many detected feature points that does not match to the other image, it is a good idea to use a threshold as well. Again, choosing this threshold might be hard.

Therefore, in [60] it is suggested that one instead looks at the ratio of the distance to the nearest neighbour and the second nearest neighbour. A false match is likely to have a similar distance to several points, while a correct match more likely has a significantly larger distance to the second nearest neighbour compared to the first. This would mean that two feature points $(\mathbf{x}_k, \mathbf{f}_k)$ and $(\mathbf{y}_l, \mathbf{g}_l)$ match if

$$\frac{\min_{j, j \neq l} \|\mathbf{f}_k - \mathbf{g}_j\|}{\|\mathbf{f}_k - \mathbf{g}_l\|} < t_2, \tag{5.3}$$

where t_2 again is some pre-set threshold.

Another way to establish matches could be to first find the nearest neighbour of each $\mathbf{x}_i$ in $\mathbf{Y}$, and then find the nearest neighbour for each $\mathbf{y}_j$ in $\mathbf{X}$ and see if the closest points agree. This means that if

$$\begin{cases} \text{argmin}_j \|\mathbf{f}_k - \mathbf{g}_j\| = l, \\ \text{argmin}_i \|\mathbf{f}_i - \mathbf{g}_l\| = k, \end{cases} \tag{5.4}$$

then $(\mathbf{x}_k, \mathbf{f}_k)$ and $(\mathbf{y}_l, \mathbf{g}_l)$ are a match.

The easiest way to search for matches and nearest neighbours is to compute the distance between all pairwise points. This can, however, be very time consuming. For examples of more efficient search methods, see [90].

Once tentative matches are established using either of the methods above, it is common to use geometry to find which of these matches that are actual matches, or inliers, and which

are outliers. This can be done in a RANSAC framework, see Section 2.6. The geometric alignment will be discussed further in the next section, but the idea is that we choose the minimum number of required point matches to compute some chosen transformation. Using these, the transformation parameters are calculated, whereupon all points in one of the images are transformed accordingly. Finally, using some threshold the consensus set is computed. This is iterated until the largest consensus set is found – the set of points which are inliers. The rest of the points will be outliers, or false matches. There are also methods for finding a good inlier set that are built on deep networks [70, 104, 106].

Finally, to evaluate the used method or threshold, one can for example count the number of *true positives* (TP), *true negatives* (TN), *false positives* (FP) and *false negatives* (FN). The true positives are the true matches that are correctly estimated as matches and the false positives are points that are not matches but that are estimated to be. True and false negatives are defined accordingly. Using these, we can define the *true positive rate* (TPR) and the *false positive rate* (FPR)

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \qquad \text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}. \tag{5.5}$$

The desire is to reach a high TPR and a low FPR, something that is reflected in the *receiver operating characteristic* (ROC) curve, where the TPR is plotted against the FPR. The curve is created by evaluating the rates for several different threshold, and a good method will have a curve that lies close to the upper left corner. All this does, of course, require that there is a ground truth for the inlier set.

5.1.1 Features in 3D

The image features are enough to create 3D models, but to merge different point clouds – or to just create maps – we need a way to describe points in 3D. One way to do this, which is also what we have been using in the thesis, is to bring the 2D features into 3D. Each 3D point is a reconstruction of a number of 2D points from different images. Each of these points has a descriptor and all of them look similar, but are not necessarily the same. In Paper VIII we let the 3D point have the mode of the ORB features for the corresponding 2D points and match them according to Equation (5.4). Another way to do this is to let the whole set of descriptors describe the 3D point, either by saving all of them or, for example, the mean and the standard deviation. There are also descriptors that are developed to work on point clouds, for example [37, 40, 102].

The matching of feature points between point clouds can be done in a similar fashion as for images, described above. The deep learning methods might, however, not work, but the other methods will – with the exception that $\mathbf{X}$ and $\mathbf{Y}$ above will be replaced by the point

clouds and the geometric alignment in the RANSAC loop will have to be adjusted. Again, there are learning method for this that work directly on the point clouds, such as [77, 101].

5.2 Point Cloud Registration

Point cloud registration refers to the alignment of point clouds, typically using a rigid transformation or a similarity transformation. The problem can also be referred to as the *absolute orientation problem*. The problem can be solved either using known correspondences between the point clouds, or without such correspondences. There are many different algorithms for point cloud registration, but we will only over a couple in this introduction.

For the registration problem we have two point clouds $\mathcal{P}_1 = \{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$ and $\mathcal{P}_2 = \{\mathbf{q}_1, \ldots, \mathbf{q}_n\}$ with points in $\mathbb{R}^3$ and N point correspondences $(\mathbf{p}_{k_i}, \mathbf{q}_{l_i})$, $i = 1, 2, \ldots, N$. The subscript indices k_i and l_i are due to potentially different orderings within the two point clouds. Now, we seek to find a scale parameter $s \in \mathbb{R}$, a translation vector $\mathbf{t} \in \mathbb{R}^3$ and a rotation matrix $\mathbf{R} \in \mathrm{SO}^3$ that minimises

$$\sum_{i=1}^{N} \|\mathbf{p}_{k_i} - (s\mathbf{R}\mathbf{q}_{l_i} + \mathbf{t})\|^2, \tag{5.6}$$

that is, we seek the least squares solution. Here, we assume that the measurement errors are Gaussian with mean zero. This problem can be solved using *Procrustes analysis* [38, 39]. For this, we begin by defining the centroids of the point sets as

$$\bar{\mathbf{p}} = \frac{\sum_{i=1}^{N} \mathbf{p}_{k_i}}{N}, \qquad \bar{\mathbf{q}} = \frac{\sum_{i=1}^{N} \mathbf{q}_{l_i}}{N}, \tag{5.7}$$

and the points translated by these centroids as

$$\mathbf{p}'_i = \mathbf{p}_{k_i} - \bar{\mathbf{p}}, \qquad \mathbf{q}'_i = \mathbf{q}_{l_i} - \bar{\mathbf{q}}. \tag{5.8}$$

These new points are such that $\sum_{i=1}^{N} \mathbf{p}'_i = \sum_{i=1}^{N} \mathbf{q}'_i = 0$. Now, the covariance of the point sets is

$$\mathbf{H} = \sum_{i=1}^{N} \mathbf{p}'_i (\mathbf{q}'_i)^T, \tag{5.9}$$

and the optimal rotation matrix is the closest orthogonal matrix to $\mathbf{H}$. Hence, if the SVD of $\mathbf{H}$ is $\mathbf{H} = \mathbf{U}\Sigma\mathbf{V}^T$, the optimal rotation matrix is given by

$$\mathbf{R} = \mathbf{U}\mathbf{V}^T. \tag{5.10}$$

Finding this rotation matrix – or rather an orthogonal matrix $\mathbf{Q}$ that minimises $\sum_{i=1}^{N} \|\mathbf{p}_{k_i} - \mathbf{Q}\mathbf{q}_{l_i}\|^2$ – is known as the orthogonal Procrustes problem [80]. To ascertain that $\mathbf{R}$ is a rotation and not a reflection, that is, that $\det(\mathbf{R}) = 1$, we can instead let [110]

$$\mathbf{R} = \mathbf{U} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \det(\mathbf{U}\mathbf{V}^T) \end{bmatrix} \mathbf{V}^T. \tag{5.11}$$

Once the rotation matrix is obtained the scaling factor is given as

$$s = \frac{\sum_{i=1}^{N} (\mathbf{p}'_i)^T \mathbf{R}\mathbf{q}'_i}{\sum_{i=1}^{N} \|\mathbf{p}'_i\|^2}, \tag{5.12}$$

and the translation vector is

$$\mathbf{t} = \bar{\mathbf{p}} - s\mathbf{R}\bar{\mathbf{q}}. \tag{5.13}$$

To use Procrustes analysis, at least three point correspondences are needed, since there are seven degrees of freedom to solve for. It is also common that the scale is known beforehand, such that only the rotation and translation are sought.

If point correspondences within the point clouds are unknown, the data association can be solved together with the registration in an iterative manner. The first step is then to associate each point in $\mathcal{P}_2$ with the nearest point in $\mathcal{P}_1$. Then, the registration step is performed using these point correspondences. After this, the data association is re-done, again using nearest neighbour, whereupon a new registration step is performed, and so on. This is done until some termination criteria is reached. This is known as the *iterative closest point* (ICP) algorithm [9]. In general, the algorithm assumes that the scale of the two point clouds is the same. There are, however, variants where a scaling factor is included, as in the Procrustes registration above [110]. There are also several other variants of the original ICP algorithm [76].

Once the point clouds have been aligned, the gauge freedom for the whole map still remains. This means that the maps can be scaled, rotated and translated without any change of the error.

5.3 Merging Point Clouds

Registration of point clouds is a good way to add information to an existing map, for example by adding local maps of smaller environments to a global map that captures a larger area. However, the final map will contain duplicates of the points that matched between the local maps, something that might not be desirable. To begin with, these duplicates will

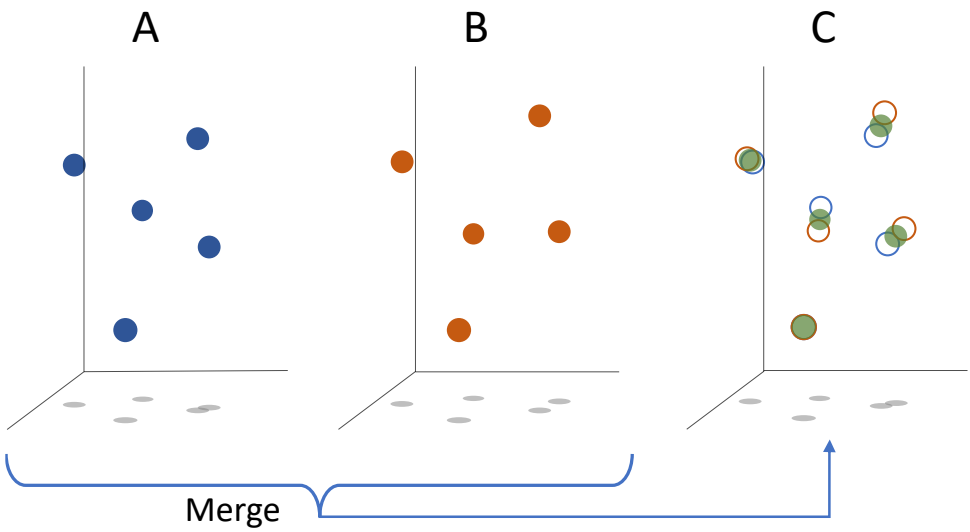


Figure 5.3: A and B show two different map representations – in blue and red, respectively – of the same environment. However, the corresponding feature points are not exactly the same. In C these are merged into one global map, in green. The positions of the original feature points are shown by contours.

require a higher memory capacity. Also, it is not guaranteed that matched points will end up at the exact same position. It is actually rather unlikely, as this would mean that there is no noise or other disturbances in the system, such that the cost function in Equation (5.6) is minimised to zero. However, if the points are a true match, they should be at the same position. With this as a motivation, it can be a good idea not only to align the maps, but rather to fuse them. A small map fusion example, to illustrate the idea, can be seen in Figure 5.3.

One straightforward way to add several maps would be to make a new estimation of the environment, using all the data that was used to obtain each of the individual maps. In the case of computer vision, this would correspond to re-estimating the parameters and then bundle over all 3D points and all camera matrices for each of the local maps at once. In terms of accuracy, this is a good method for merging, as all error terms are minimised simultaneously. However, this process could be time consuming and computationally heavy, since the number of images involved grows quickly.

Another, more efficient, way to fuse maps could be to first co-register the maps and then merge the matched points in some way, for example by replacing the pair by the average point or simply neglecting one of the points [16]. To improve the map further, this can be followed by (local) non-linear optimisation. One advantage with allowing some optimisation and not only rigid registration is that inherent drift and local errors can be corrected for.

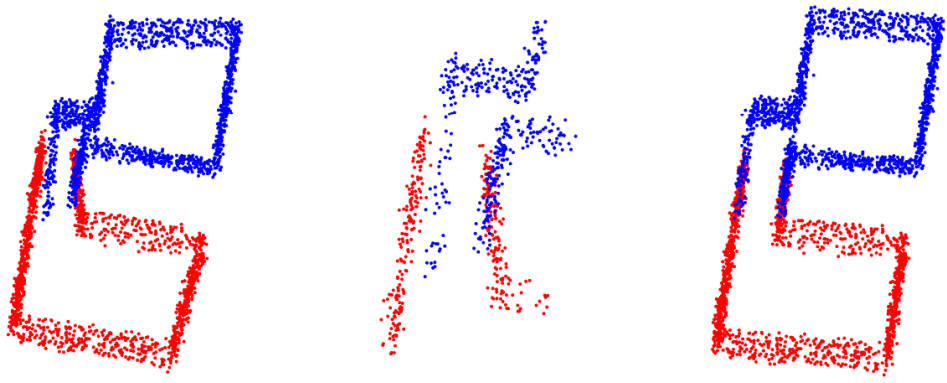


Figure 5.4: The figure shows a motivating example, where the red and blue maps to the left should be matched. Since the overlapping area contains drift, a rigid matching method will not be able to find all the matches, see the zoomed in image in the middle. Allowing for statistical deformations, such matches can be found and the errors can be corrected, as in the right plot.

Some fields for which map merging is useful are loop closure and collaborative SLAM [1, 105, 109]. In many cases, the collaborative SLAM systems are initialised such that all cameras view the same scene – something that simplifies the problem [111]. Also, the actual merge is rarely the focus of these papers and often the process is not well described. There are, however, examples of papers where this is not the case. In [16] the multi-session mapping is brought up as one of the advantages of their SLAM system and the merging is given some focus. The matching is, in similarity with many other papers, based on keyframes and the merging is done by erasing duplicates from the local maps, followed by local bundle adjustment over the concerned keyframes. Other methods that use keyframes – and more specifically bag-of-words for these – for detection of loop closure or map matching are [31, 63, 89].

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 [1, 11] and that a key step in the merging is to extract stable point features [105].

Map merging has been one of the focus areas of this thesis. One of the aims has been to find efficient merging methods, where no images – not even keyframes – have to be saved and the bundle over all reprojection errors can be avoided. A key idea here has been to divide the parameters in one set of *main parameters* and one of *auxiliary parameters*. The main parameters are those that are of higher interest. This could for example be all map points, or a subset of the map points that we know are static, are likely to match over several mappings or generally represent the environment in a good way. The rest of the parameters – that is, the rest of the 3D points and the cameras – belong to the auxiliary parameters. This division is introduced in Paper VI. Once this division is done, we only explicitly optimise over the main parameters. This reduces the memory footprint and simplifies the cost function.

The second aim for the fusion papers of this thesis has been to allow the map points to not only transform rigidly, but also to move in the directions where the local maps are most uncertain. This does not necessarily give a non-rigid transformation, but rather a rigid transformation with statistical deformations. Figure 5.4 shows an example of the type of errors that we try to avoid by allowing these statistical deformations. The two maps that are to be merged both contain some inherent drift and therefore a fully rigid transformation will never result in a perfect match. Map fusion is treated in Papers VI, VII and VIII of this thesis.

Chapter 6

Conclusions

This thesis has a focus on mapping and merging using sound and vision. It is built on eight conference papers, and as a finish to this introductory part of the thesis, we will include a summary and discussion on some of the main contributions of these papers. The papers can roughly be divided into three parts: Paper I which has a focus on refining TDOA measurements; Papers II–V which treat self-calibration using sound and ultra wide-band; and Papers VI–VIII which have a focus on map merging.

6.1 Paper I: Precise Measurements

The first paper covers the earliest step in the process treated in this thesis, namely to achieve accurate TDOA measurements. While the paper does not explicitly handle mapping and self-calibration, this pre-processing step is of importance for the mapping results as well. In the paper we work with sound signals, but the methods are applicable for any one-dimensional signal modality. The paper is inspired by [4] where scale space smoothing and stochastic analysis are used in similar settings but for images, and by [107], where TDOA was studied on a subsample level. In the latter of these, it was empirically shown that the subsample refinements improved the calibration process.

Our paper continues on the same track, but we also present an analysis of the limitations of the methods by investigating the uncertainties in the estimates. Rather than using cross-correlation to find the TDOA measurements, we formulate this as a minimisation of a loss function, see Equation (3.17). While these two formulations are analogous if only a time delay is estimated, our formulation allows for more variations in the model and makes it easier to analyse the uncertainty. We then present explicit formulae for the standard deviation of these time delay estimates.

The second contribution of the paper is the estimation of a Doppler factor, together with uncertainty estimates thereof, and an amplitude, see Equation (3.16) and Figure 3.2. The Doppler factor becomes relevant for signals where the sound source is moving, and despite noisy measurements we show that there are relevant information to gain even when the sound source is moving slowly. We show on synthetic data that the estimation of the Doppler factor also improves the estimate of the time delay, if the sound source has moved. However, it seems like our theoretical formula underestimates the standard deviation for the Doppler factor.

Future Work

One topic for further research would be to look into why the standard deviation of the Doppler factor is underestimated. It would also be interesting to see if there are other use cases for the Doppler and amplitude estimates rather than to only improve on the time delay.

6.2 Papers II–V: Self-Calibration Using One-Dimensional Signals

The topic in Paper I gives a good transition into Papers II–V, where the measurements are used for robust automatic calibration – that is, to find both sender and receiver positions, as we discussed in Section 4.2. Mostly in these papers, we assume that the distance measures are given and in the experiments we often use cross-correlation (Equation (2.20)) or GCC-PHAT (Equation (3.22)) to compute them. The four papers cover different parts of the calibration process, but are all closely connected. The topic has also been studied at the Centre for Mathematical Sciences at Lund University for several years, and several PhD theses include work that has lead up to these papers [6, 14, 45, 52, 82, 87]. In all Papers II–V the stratified solution approach explained in Section 4.2.2 is used. Furthermore, all papers have a focus on robust estimation and use a RANSAC approach (Section 2.6) to achieve this, both to find a good solution and a large inlier set. We have also developed minimal solvers in each of the papers. These have been created using the techniques in Section 2.4 and an automatic solver generator [53]. The minimal solvers have been used to initialise a solution, whereupon this has been extended to more rows and columns, according to Section 4.2.3.

Paper II

Paper II treats the COTDOA problem – TDOA with a constant offset – from Equation (3.8). This case arises when the emission of the sound events are repetitive with a known period, and only one offset has to be estimated, in comparison to the original TDOA problem where there is one offset per sender. In the paper, we study the (5r/5s) problem including five receivers and five senders and provide a fast minimal solver for the first step in the stratified approach. This step involves solving for the relaxed parameters $\boldsymbol{\theta}_2 = \{\mathbf{U}, \mathbf{V}, \mathbf{b}, \mathbf{a}, o\}$, see Section 4.2.2. Specifically, we express a 4th degree polynomial in terms of a determinant, using the constraint that the double compaction matrix from Equation (4.13) has rank 3. When this polynomial is solved we get four possible solutions for the offset o. Each of these possible solutions then give a possible solution for $\boldsymbol{\theta}_2$. The upgrade to the original parameters $\boldsymbol{\theta}_1 = \{\mathbf{R}, \mathbf{S}, o\}$ is then done using techniques presented in [47, 54].

Using the presented solver, the step of calculating o and the rest of $\boldsymbol{\theta}_2$ is significantly faster than that of upgrading to $\boldsymbol{\theta}_1$. Therefore, we suggest that local optimisation is done for the relaxed problem before upgrading the solution (see Section 4.2.4). Since the minimal solver is fast we can allow for many RANSAC iterations. The solver also shows good numerical stability.

Paper IV

Paper IV also covers the first step of the two-tiered approach, but for the regular TDOA problem in Equation (3.3). In the paper, we improve on three non-linear minimal solvers that were previously presented in [46], for the cases (5r/6s), (7r/6s) and (6r/8s) – where (xr/ys) again refers to a system with x receivers and y senders. The new solvers use the rank 3 constraint of the compaction matrix (Equation (4.13)), which is expressed as all minors of the matrix of order 4 being zero. These minors are expressed in lower order minors using Laplace expansion, which makes some common sub-expressions cancel. This leads to a smaller system of equations which can be solved using action matrix methods (Section 2.4). The simplification makes the solvers significantly faster and more memory efficient solvers than the solvers in [46]. We also consider two linear cases from the same paper, namely (7r/4s) and (9r/5s).

We suggest how to use these five solvers in a full RANSAC system. To solve the self-calibration we randomly choose one of the five presented solvers and a suitable subset of the measurements. The offsets in $\mathbf{o}$ are found using the solver, and then we compute the rest of the relaxed parameters using the compaction matrix, by extending to more rows and columns according to Section 4.2.3. To improve the results further, we occasionally do local optimisation over $\boldsymbol{\theta}_2$. At this stage we minimise directly over the measurement

errors, according to the error function in Equation (4.17). Thereafter the upgrade is done using the methods presented in Paper III (see next paragraph) whereupon the receiver and sender positions are re-estimated using trilateration and multilateration (see Section 4.1) to improve the results. The results show that the system is robust to outliers and missing data. The linear solvers are more numerically stable than the non-linear solvers, but since there might be cases where there are not enough receivers to use, for example, the (9r/5s) solver, all solvers could be of use. The code for the paper is available online[1].

Paper III

Following these two papers, Paper III has a focus on the second step of the stratified method (Section 4.2.2). Again, we present several minimal solvers, this time for the upgrade from the relaxed $\boldsymbol{\theta}_2$ to $\boldsymbol{\theta}_1$. Since the offsets are determined in the first step, the methods in this paper are applicable to all the cases of time of arrival (TOA); time-difference of arrival (TDOA); constant offset TDOA (COTDOA); and uncalibrated TDOA (UTDOA), see Equations (3.1), (3.3), (3.8) and (3.9), respectively. To compensate for the larger gauge freedom in the relaxed problem we add conditions on the relaxed solution, which we meet by translating the parameters in $\boldsymbol{\theta}_2$. We also suggest a new, more general, formulation of the double compaction matrix – namely the one in Equation(4.13) where the computations do not depend on a certain column of the distance matrix. We then decompose the double compaction matrix and parameterise the sender and receiver positions in terms of two reference points and a transformation matrix. This leads to a system of equations with nine unknowns and we derive equations of three types, one of which is linear and two non-linear. There are in total 19 minimal configurations and we present solvers for ten of these; one which only builds on linear equations and the rest on increasingly amount of non-linear equations. The systems of equations are solved for using action matrix methods and Cholesky factorisation, see Section 2.3.1 and 2.4 .

When we solve for the upgrade we follow the same solution scheme as [54]. However, we present smaller templates for setting up the action matrices for some of the problems that they solved and extend the number of solvers, as they only studied certain receiver/sender configurations. The linear solvers exhibit the best numerical stability for the fully spanned cases, but are more sensitive to degenerate configurations. We also show that better results can be achieved by optimising over the upgrade parameters, which we used to parameterise $\mathbf{R}$ and $\mathbf{S}$. The code for the paper is available online[2].

[1]The code can be found at `https://github.com/martinkjlarsson/tdoa-self-calibration`.
[2]The code is available at `https://github.com/martinkjlarsson/upgrade-methods`.

Paper V

The fourth paper in this group, Paper V does not focus on the stratified solution, but rather on improving solutions from self-calibration – that is, to improve on the receiver and sender positions. If the sound source has been moving there are often many more sender positions than receivers. In that case it is common that only a partial solution for the sound source path can be found, such that the sender positions are found for some time steps but not for others. In this paper we present minimal solvers for multilateration (see Section 4.1) using TDOA measurements to pairs of microphones that improve on the sound source position. We also present a framework for how to use these solvers for robust improvement on all the node positions – sender as well as receiver positions. Furthermore, we provide a new dataset for TDOA multilateration[3].

One difference compared to the previous papers is that in this paper we assume that the receiver positions are known, when we estimate the sound source positions using multilateration. We do this improvement by going back to the putative TDOA measurements, for example the peaks of the GCC-PHAT from Equation (3.22). That the measurements are putative means that we sometimes do not know which correlation peak is the strongest and therefore sometimes use several peaks. Previous methods for this has required that all time-differences are given to the same microphone, while we extend this so that any three measurements from the TDOA matrix (see Section 3.1.1) can be used. Using an automatic solver generator [53] we produce a solver, which together with the problem formulation is a contribution of the paper. Furthermore, we consider the inclusion of smoothness priors on the sound source path, and we also include local optimisation over both sound source and receiver locations. The code for the paper is available online[4].

Summary and Future Work

An overview of the whole calibration and improvement system can be seen in Figure 6.1. The input to the system, Figure 6.1a, is sound recordings, or some other type of one-dimensional signals. The step between Figure 6.1a and 6.1b consists of finding putative TDOA estimates and the step from Figure 6.1b to 6.1c to refine these. Paper I would be positioned here, going from sensor data to TDOA measurements. All of Papers II, IV and III cover the step from Figure 6.1c to 6.1d, where TDOA measurements are used to estimate sender and receiver positions. This step could have been divided into two, for the relaxed solution and the upgrade. Finally, Paper V uses both the node positions in Figure 6.1d and the original, putative TDOA measures in Figure 6.1b to improve on the solution, especially for the sender path. The result is a refined solution, as in Figure 6.1e.

[3]The dataset can be found at `https://vision.maths.lth.se/sfsdb/`.

[4]Link to the code `https://github.com/kalleastrom/StructureFromSound`.

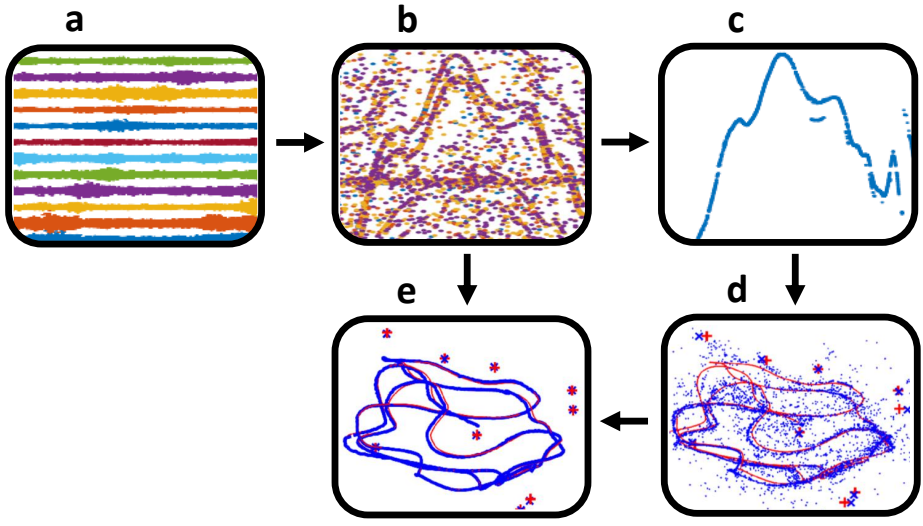


Figure 6.1: The figure shows an overview of the self-calibration system and the different parts that have been included in this thesis. In (a) the raw sound data is shown, in (b) noisy TDOA measurements and (c) refined measurements. The figure in (d) shows noisy sender and receiver estimates and (e) a refinement of these. The different papers in this theses correspond to the arrows.

One future goal for this set of papers – covering robust self-calibration – is to develop a toolbox, where each combination of calibration type (TOA, TDOA, COTDOA, UTDOA, see Section 3.1.1) and each reasonable dimensionality of the senders and receivers is covered. For this, we want to find as many common formulations as possible for the different cases. The stratified solution approach is one step on the way here, but there might be more to be found. Each paper will contribute as one (or several) tool in the toolbox, and there are still minimal cases to consider before every case is covered. For example, there are many unanswered questions concerning the UTDOA case (Equation (3.9)). The minimal cases get much larger as there are offsets for both senders and receivers.

Another interesting topic for future research is to develop good methods for refining the putative TDOA measurements, that is the step between Figure 6.1b to 6.1c. Given GCC-PHAT values we wish to choose those that give high peaks, but if there has been a moving sound source, we also want the chosen values to be smooth over time. It is possible that this problem could be solved using learning based methods.

6.3 Papers VI–VIII: Map Merging

The last group of papers concern map matching and merging (explained in Chapter 5) and requires that maps have already been created, for example using the techniques described in the previous papers, or using some SfM or SLAM pipeline, see Section 4.5. Papers VI and VII treat the actual merging process, while Paper VIII concerns the step before, where point matches are ascertained. Nevertheless, the three papers have much in common and follow the same idea. The motivation behind the papers is to establish a technique for fusion of individual, local maps, into a more accurate, global, map, without having to re-estimate the map from scratch using bundle adjustment over all reprojection errors.

Paper VI

The problem is introduced in Paper VI, where we present a method for efficient map merging with a small memory footprint. The paper is mainly focused on sound data, but also covers some vision. One of the key ideas for the compact map format is to divide the parameters – that is the senders and receivers or the cameras and 3D points – into main parameters and auxiliary parameters, as described in Section 5.3. By dividing the Jacobian accordingly, we can use first order Taylor expansion to get an approximate expression for how the residuals are affected by a change in the main parameters. The auxiliary parameters will follow this change, but they are not the focus and will not explicitly be optimised over. By this, we can express how the error changes if we change the map, using only the error in the optimal point; the main map points; and a triangular matrix obtained from QR decomposition of a Jacobian. The error can for example be the one from Equations (4.16) or (4.49). One big advantage here is that the triangular matrix that we get from QR decomposition encodes the same information as the Jacobian, but contains much fewer values, see Equation (2.32).

For the actual fusion, we assume that the maps have been aligned, so that they are in the same coordinate system, see Section 5.2. Using the linearisation of the residual and the compact representation of the map, the merge can be solved linearly. This makes the merge very efficient. Through this we keep most of the information from the individual bundles, but we focus on the map points that are of particular interest – that is the main parameters. Also, the problem does not scale even if more auxiliary parameters are added; for example, it does not matter how many sender positions that are used to create the local maps, if the number of receivers is the same (and the receivers are the main parameters).

Furthermore, if the errors (Equation (4.16) or (4.49)) from the individual bundles are assumed to be Gaussian with mean zero, we show how the total error in the individual and the merged maps can be used to discover whether any changes have occurred in the

scene between the mapping occasions. It turns out the the error should increase according to a gamma distribution when maps are merged. If the increased error is far from this distribution, one can suspect that something has changed. This could for example be used as a hypothesis test to remove inconsistent parts of the maps.

Paper VII

The assumption that the maps are aligned is a disadvantage of Paper VI. However, we address this issue in Paper VII. There, we use the same compact map representation as in the previous paper, but we develop the merging algorithm to work for maps represented in different coordinate systems. This means that we solve the registration problem (Section 5.2) and the merging problem (Section 5.3) simultaneously. In this paper we have a focus on image data, with maps originating from SLAM or SfM, although the method works for other signal modalities as well.

In Paper VI, described in the previous section, all points in the global map have to be visible in all the local maps. This is generalised so that the global map can contain points that are visible in one or more of the local maps, and that no points have to be seen in all maps. We represent this by including a projection from the space of the global map to those of the main parameters of the local maps. Furthermore, we include a similarity transformation for each local map, that transforms between the local and the global coordinate system, see Equation (5.6). Since the change of coordinate system is non-linear, we can no longer solve the merging linearly. Therefore, we collect the individual residuals and optimise over the global map points and the transformation matrices. To ensure that the linearisations for the local maps are still valid, we also add extra penalties by changing the nullspace of the triangular matrices representing the Jacobians.

Even though we perform bundle adjustment (described in Section 2.5.2) to optimise the global map, it will be much more efficient than a bundle over all 3D points and cameras, as we have reduced the number of parameters to optimise, as well as the number of residuals. In the original error function all reprojection errors in all cameras are included, while our cost function only includes the errors of the main 3D points. Except for these contributions, we also updated the formula for the hypothesis test, used for change detection, to work for the new settings.

Paper VIII

In the two merging papers described we assume that point matches between the maps were given. The process of finding such matches is the subject of Paper VIII. Given a set of tentative matches between two point clouds we develop methods to refine these matches

and find a reliable inlier set, see Section 5.1. In this match, we allow the points to modify in the direction of the largest modes of variation – that is, in the direction where the local map is most uncertain and the error function is least affected, see Section 2.3.1. Through this, errors that occurred in the original SLAM can be corrected for.

We formulate and present solvers for the minimal problems of matching both three and four point pairs, both for similarity and for Euclidean transforms. The three-point-matching problem includes two and three modes, respectively, and the four-point-matching problem five and six modes, respectively. Furthermore, we present a solver for matching of duplicate points within maps, which can be used to solve loop closure problems (see Chapter 5). The systems of equations for matching between maps are solved using methods from algebraic geometry, see Section 2.4.2. We use an automatic solver generator to produce these solvers, but adjust the solvers manually to make the calculations more simple and effective. The solvers are available online[5].

Finally, we suggest how the minimal solvers can be utilised in a RANSAC framework (Section 2.6). We also use the resulting matches as input to the merging algorithm presented in Papers VI and VII. For this to work, we adjust both the fusion method and the hypothesis test to allow for matches within maps. The code developed throughout these merging papers is available online[6].

Future Work

We have had several discussions about how to formulate the problems concerning map merging and believe that there are still many ways to explore and expand the papers on the subject included in this thesis. One potential topic for future research could be to make the methods work in a larger setting, where the hypothesis test (that is the method we used for change detection in Paper VI) could be used for increased robustness, by dividing uncertain maps and adding one part at a time. If the maps are large, this would also require some development of the code, to speed up the computations for larger sets of data. It would also be interesting to find ways to express the compressed map representation for a smaller set of main parameters from the compressed representation for a larger set of parameters. This would give more freedom in the division of the parameters, as it could then be re-made without having to re-compute the original bundles. It would also be interesting to allow for cameras to be included in the main parameters.

Another investigation for the future could be to combine these geometric models with modern deep learning techniques. If we could determine the semantics for the 3D points this could be used to decide which parts that are more likely to be static and therefore are

[5]Solvers at `https://github.com/gabrielleflood/statistical-mapmatching-minsolv`.
[6]The code can be found at `https://github.com/gabrielleflood/mapmerging`.

good to use as main parameters. One could also try to use both semantics, geometry and the original 2D features for point matching, in order to obtain a more robust 3D feature.

References

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

[2] X. Anguera, C. Wooters, and J. Hernando. Acoustic beamforming for speaker diarization of meetings. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(7):2011–2022, 2007.

[3] X. Anguera, S. Bozonnet, N. Evans, C. Fredouille, G. Friedland, and O. Vinyals. Speaker diarization: A review of recent research. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(2):356–370, 2012.

[4] K. Åström and A. Heyden. Stochastic analysis of image acquisition, interpolation and scale-space smoothing. *Advances in Applied Probability*, 31(4):855–894, 1999.

[5] K. Åström, M. Larsson, G. Flood, and M. Oskarsson. Extension of time-difference-of-arrival self calibration solutions using robust multilateration. In *29th European Signal Processing Conference (EUSIPCO)*. IEEE, 2021.

[6] K. Batstone. *Computer Vision without Vision: Methods and Applications of Radio and Audio Based SLAM*. PhD thesis, Lund University, 2020.

[7] K. Batstone, M. Oskarsson, and K. Åström. Robust time-of-arrival self calibration with missing data and outliers. In *2016 24th European Signal Processing Conference (EUSIPCO)*, pages 2370–2374. IEEE, 2016.

[8] 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.

[9] 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.

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

[11] 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.

[12] M. Brandstein, J. Adcock, and H. Silverman. A closed-form location estimator for use with room environment microphone arrays. *IEEE Transactions on Speech and Audio Processing*, 5(1):45 –50, 1997.

[13] M. S. Brandstein and H. F. Silverman. A robust method for speech signal time-delay estimation in reverberant rooms. In *1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 1, pages 375–378. IEEE, 1997.

[14] S. Burgess. *Minimal Problems and Applications in TOA and TDOA Localization*. PhD thesis, Lund University, 2016.

[15] 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.

[16] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap SLAM. *IEEE Transactions on Robotics*, pages 1–17, 2021.

[17] O. Chum, J. Matas, and J. Kittler. Locally optimized RANSAC. In *Joint Pattern Recognition Symposium*, pages 236–243. Springer, 2003.

[18] A. Cirillo, R. Parisi, and A. Uncini. Sound mapping in reverberant rooms by a robust direct method. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 285 –288. IEEE, 2008.

[19] M. Cobos, A. Marti, and J. Lopez. A modified SRP-PHAT functional for robust real-time sound source localization with scalable spatial sampling. *IEEE Signal Processing Letters*, 18(1):71 –74, 2011.

[20] 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.

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

[22] M. Crocco, A. Del Bue, M. Bustreo, and V. Murino. A closed form solution to the microphone position self-calibration problem. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2597–2600. IEEE, 2012.

[23] M. Csorba. *Simultaneous localisation and map building*. PhD thesis, University of Oxford, 1997.

[24] M. Demazure. *Sur deux problemes de reconstruction*. PhD thesis, Inria, 1988.

[25] 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.

[26] H. Do, H. Silverman, and Y. Yu. A real-time SRP-PHAT source location implementation using stochastic region contraction(SRC) on a large-aperture microphone array. In *2007 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 1, pages 121–124. IEEE, 2007.

[27] H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

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

[29] 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.

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

[31] X. Gao, R. Wang, N. Demmel, and D. 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.

[32] C. F. Gauss. *Theoria Motus Corporum Coelestium in sectionibus conicis solem ambientium*. Göttingen, 1809.

[33] S. K. Ghosh. *Analytical photogrammetry*. Pergamon Press, second edition edition, 1988.

[34] C. Harris, M. Stephens, et al. A combined corner and edge detector. *Alvey Vision Conference*, 15(50):10–5244, 1988.

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

[36] M. H. Hayes. *Statistical digital signal processing and modeling*. John Wiley & Sons, 1996.

[37] L. He, X. Wang, and H. Zhang. M2DP: A novel 3D point cloud descriptor and its application in loop closure detection. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 231–237. IEEE, 2016.

[38] 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.

[39] 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.

[40] J. Huang and S. You. Point cloud matching based on 3D self-similarity. In *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 41–48. IEEE, 2012.

[41] G. Klein and D. Murray. Parallel tracking and mapping for small AR workspaces. In *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234. IEEE, 2007.

[42] C. Knapp and G. Carter. The generalized correlation method for estimation of time delay. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 24(4):320–327, 1976.

[43] S. Korman and R. Litman. Latent RANSAC. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6693–6702. IEEE, 2018.

[44] V. A. Kotelnikov. On the transmission capacity of ether and wire in electrocommunications. *Material for the First All-Union Conference on Questions of Communication, Izd. Red. Upr. Svyazi RKKA (in Russian). (English translation, PDF)*, 1933.

[45] Y. Kuang. *Polynomial Solvers for Geometric Problems – Applications in Computer Vision and Sensor Networks*. PhD thesis, Lund University, 2014.

[46] Y. Kuang and K. Åström. Stratified sensor network self-calibration from TDOA measurements. In *21st European Signal Processing Conference (EUSIPCO)*, pages 1–5. IEEE, 2013.

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

[48] Z. Kukelova, M. Bujnak, and T. Pajdla. Automatic generator of minimal problem solvers. In *European Conference on Computer Vision (ECCV)*, pages 302–315. Springer, 2008.

[49] M. Larsson, G. Flood, M. Oskarsson, and K. Åström. Upgrade methods for stratified sensor network self-calibration. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4851–4855. IEEE, 2020.

[50] M. Larsson, G. Flood, M. Oskarsson, and K. Åström. Fast and robust stratified self-calibration using time-difference-of-arrival measurements. In *2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4640–4644. IEEE, 2021.

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

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

[53] V. Larsson, K. Astrom, 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.

[54] V. Larsson, K. Åström, and M. Oskarsson. Polynomial solvers for saturated ideals. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2288–2297. IEEE, 2017.

[55] A. M. Legendre. *Nouvelles méthodes pour la détermination des orbites des comètes*. F. Didot, 1805.

[56] K. Levenberg. A method for the solution of certain non-linear problems in least squares. *Quarterly of Applied Mathematics*, 2(2):164–168, 1944.

[57] T. Lindeberg. Scale-space theory: A basic tool for analyzing structures at different scales. *Journal of Applied Statistics*, 21(1-2):225–270, 1994.

[58] G. Lindgren, H. Rootzén, and M. Sandsten. *Stationary Stochastic Processes for Scientists and Engineers*. CRC Press, 2013.

[59] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293(5828):133–135, 1981.

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

[61] 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.

[62] R. Mur-Artal and J. D. Tardós. ORB-SLAM2: An open-source SLAM system for monocular, stereo, and RGB-D cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

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

[64] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. DTAM: Dense tracking and mapping in real-time. In *2011 International Conference on Computer Vision (ICCV)*, pages 2320–2327. IEEE, 2011.

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

[66] D. Nistér, O. Naroditsky, and J. Bergen. Visual odometry. In *2004 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 1. IEEE, 2004.

[67] H. Nyquist. Certain topics in telegraph transmission theory. *Transactions of the American Institute of Electrical Engineers*, 47(2):617–644, 1928.

[68] C. Olsson. Lecture notes in computer vision. `https://www.maths.lth.se/matematiklth/personal/calle/datorseende19/index.html`, 2019. Accessed: 2021-10-19.

[69] 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.

[70] T. Plötz and S. Roth. Neural nearest neighbors networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, pages 1087–1098. Curran Associates, Inc., 2018.

[71] M. Pollefeys and D. Nister. Direct computation of sound and microphone locations from time-difference-of-arrival data. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2445–2448. IEEE, 2008.

[72] J. G. Proakis. *Digital signal processing: principles algorithms and applications*. Pearson Education, fourth edition edition, 2007.

[73] R. Raguram, O. Chum, M. Pollefeys, J. Matas, and J.-M. Frahm. USAC: a universal framework for random sample consensus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):2022–2038, 2013.

[74] E. Rosten, R. Porter, and T. Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(1):105–119, 2008.

[75] 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.

[76] S. Rusinkiewicz and M. Levoy. Efficient variants of the ICP algorithm. In *3rd International Conference on 3-D Digital Imaging and Modeling*, pages 145–152. IEEE, 2001.

[77] P.-E. Sarlin, D. DeTone, T. Malisiewicz, and A. Rabinovich. SuperGlue: Learning feature matching with graph neural networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4938–4947. IEEE, 2020.

[78] P. Schmuck and M. Chli. Multi-UAV collaborative monocular SLAM. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3863–3870. IEEE, 2017.

[79] 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.

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

[81] C. E. Shannon. Communication in the presence of noise. *Proceedings of the IRE*, 37 (1):10–21, 1949.

[82] Z. Simayijiang. *Applications of Signal Processing to Microphone Node Calibration and Medical Signal Classification*. PhD thesis, Lund University, 2017.

[83] 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.

[84] K. Siwiak. Ultra-wideband radio. *Encyclopedia of RF and Microwave Engineering*, 2005.

[85] N. Snavely, S. M. Seitz, and R. Szeliski. Photo tourism: exploring photo collections in 3D. *ACM Transactions on Graphics*, 25(3):835–846, 2006.

[86] 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.

[87] H. Stewénius. *Gröbner basis methods for minimal problems in computer vision*. PhD thesis, Lund University, 2005.

[88] P. Sturm and S. Ramalingam. *Camera models and fundamental concepts used in geometric computer vision*. Now Publishers Inc, 2011.

[89] S. Sun, B. Xu, Y. Sun, and Z. 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.

[90] R. Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[91] S. Thrun. Affine structure from sound. In *Advances in Neural Information Processing Systems*, volume 18, pages 1353–1360. Citeseer, 2006.

[92] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon. Bundle adjustment – a modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372. Springer Berlin Heidelberg, 1999.

[93] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[94] J. Velasco, D. Pizarro, J. Macias-Guarasa, and A. Asaei. TDOA matrices: Algebraic properties and their application to robust denoising with missing data. *IEEE Transactions on Signal Processing*, 64(20):5242–5254, 2016.

[95] Y. Wang and J. M. Solomon. Deep closest point: Learning representations for point cloud registration. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3523–3532. IEEE, 2019.

[96] E. T. Whitaker. On the functions which are represented by the expansions of the interpolation theory. In *Proc. Royal Soc. Edinburgh*, volume 35, pages 181–194, 1915.

[97] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardós. A comparison of loop closing techniques in monocular SLAM. *Robotics and Autonomous Systems*, 57(12):1188–1197, 2009.

[98] C. Wu. Towards linear-time incremental structure from motion. In *2013 International Conference on 3D Vision (3DV)*, pages 127–134. IEEE, 2013.

[99] C. Wu, S. Agarwal, B. Curless, and S. M. Seitz. Multicore bundle adjustment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3057–3064. IEEE, 2011.

[100] C. Wu et al. VisualSFM: A visual structure from motion system. `http://ccwu.me/vsfm/`, 2011. Accessed: 2021-09-30.

[101] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2021.

[102] J. Yang, Z. Cao, and Q. Zhang. A fast and robust local descriptor for 3D point cloud registration. *Information Sciences*, 346:163–179, 2016.

[103] 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.

[104] K. M. Yi, E. Trulls, Y. Ono, V. Lepetit, M. Salzmann, and P. Fua. Learning to find good correspondences. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2666–2674. IEEE, 2018.

[105] 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.

[106] C. Zhao, Z. Cao, C. Li, X. Li, and J. Yang. NM-Net: Mining reliable neighbors for robust feature correspondences. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 215–224. IEEE, 2019.

[107] S. Zhayida and K. Åström. Time difference estimation with sub-sample interpolation. *Journal of Signal Processing*, 20(6):275–282, 2016.

[108] S. Zhayida, F. Andersson, Y. Kuang, and K. Åström. An automatic system for microphone self-localization using ambient sound. In *2014 22nd European Signal Processing Conference (EUSIPCO)*, pages 954–958. IEEE, 2014.

[109] X. S. Zhou and S. 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.

[110] T. Zinßer, J. Schmidt, and H. Niemann. Point set registration with integrated scale estimation. In *International Conference on Pattern Recognition and Image Processing (PRIP)*, pages 116–119, 2005.

[111] D. Zou and P. Tan. CoSLAM: Collaborative visual SLAM in dynamic environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(2):354–366, 2013.

Scientific Publications

Paper I

Material from: G. Flood, A. Heyden and K. Åström, Stochastic Analysis of Time-Difference and Doppler Estimates for Audio Signals, *Pattern Recognition Applications and Methods*, Springer International Publishing, Cham, 2019, published 2019, Copyright Springer Nature Switzerland AG 2019.

Stochastic Analysis of Time-Difference and Doppler Estimates for Audio Signals

Gabrielle Flood, Anders Heyden and Kalle Åström

Centre for Mathematical Sciences, Lund University, Lund, Sweden

Abstract: Pairwise comparison of sound and radio signals can be used to estimate the distance between two units that send and receive signals. In a similar way it is possible to estimate differences of distances by correlating two received signals. There are essentially two groups of such methods, namely methods that are robust to noise and reverberation, but give limited precision and sub-sample refinements that are more sensitive to noise, but also give higher precision when they are initialized close to the real translation. In this paper, we present stochastic models that can explain the precision limits of such sub-sample time-difference estimates. Using these models new methods are provided for precise estimates of time-differences as well as Doppler effects. The developed methods are evaluated and verified on both synthetic and real data.

Keywords: time-difference of arrival, sub-sample methods, Doppler effect, uncertainty measure

1 Introduction

Audio and radio sensors are increasingly used in smartphones, tablet PC's, laptops and other everyday tools. They also form the core of internet-of-things, e.g. small low-power units that can run for years on batteries or use energy harvesting to run for extended periods of time. If the locations of the sensing units are known, they can be used as an ad-hoc acoustic or radio sensor network. There are several interesting cases where such sensor networks can come into use. One such application is localization, cf. [5–7, 9]. Another possible usage is beam-forming, i.e. to improve sound quality, [2]. Using a sensor network one can also determine who spoke when through speaker diarisation, [1]. If the sensor positions are unknown or if they are only known to a certain accuracy, the performance of such use-cases are inferior as is shown in [18]. It is, however, possible to perform automatic calibration, i.e. to estimate both sender and receiver positions, even without any prior information. This can be done up to a choice of coordinate system, [8, 12, 13, 19, 22], thus providing accurate sensor positions for improved use. A key component for all of these methods is the

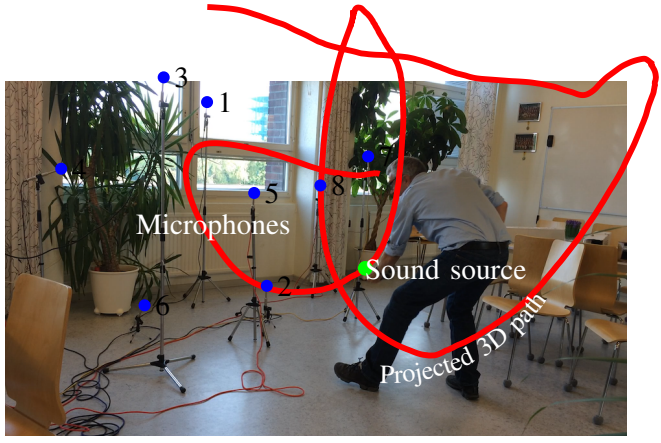


Figure 1: Precise time-difference of arrival estimation can be used for many purposes, e.g. diarization, beam-forming, positioning and anchor free node calibration. The figure illustrates its use for anchor free node calibration, sound source movement and room reconstruction. The image is taken from [10].

process of obtaining and assessing estimates of e.g. time-difference of arrival of transmitted signals as they arrive in pairs of sensors. In this paper the focus is primarily on acoustic signals, but the same principles are useful for the analysis of radio signals [4].

All of these applications depend on accurate methods to extract features from the sound (or radio) signals. The most common feature is the time-difference-of-arrival, which is then used for subsequent processing. For applications, it is important to find as precise estimates as possible. In [23] time-difference estimates were improved using sub-sample methods. It was also shown empirically that the estimates of the receiver-sender configurations were improved by this. However, no analysis of the uncertainties of the sub-sample time-differences was provided.

This paper is an extended version of [10]. The main content is thus similar. However this version has been developed and is more thorough. E.g. the derivations in Section 3.1 have been extended, a comparison between different models has been added, see Section 3 and 4.1, and the experiments on real data in Section 4.2 have been changed and improved. In addition we have also performed stochastic analysis for the real data experiments. This is presented in Section 4.2. Then follows Section 4.3 which is partly new. Furthermore, most of the figures have been updated, even if a few remain from the original paper.

The main contributions of [10] and this paper are:

- A scheme for computing time-difference estimates and for estimating the the precision of these estimates.
- A method to estimate minute Doppler effects, which is motivated by an experimental

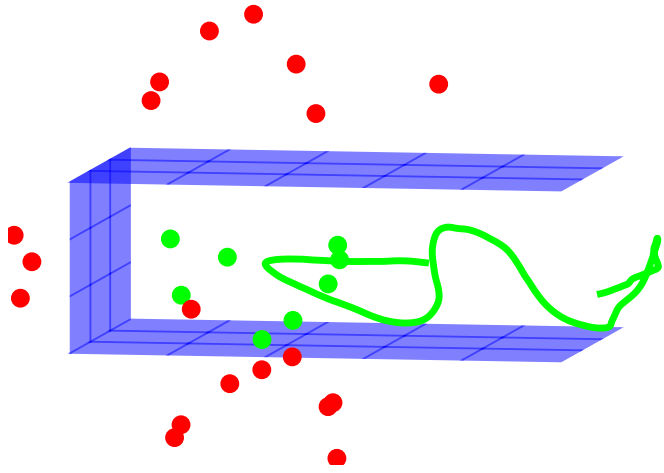


Figure 2: The figure examplifies one usage of precise time-difference of arrival estimation. The image illustrates the estimated microphone positions (dots), estimated mirrored microphone positions (dots) and sound source motion (solid curve) from Figure 1. The estimated reflective planes are also shown in the figure. These three planes correspond to the floor, the ceiling and the wall. The image is taken from [10].

comparison between the models.

- An extension of the framework to capture and estimate amplitude differences in the signals.
- An evaluation on synthetic data to evince the validity of the models and provide knowledge of when the method fails.
- An evaluation on real data which demonstrates that the estimates for time-difference, minute Doppler effects and the amplitude changes contain relevant information. This is shown for speeds as small as 0.1 m/s.

2 Modeling Paradigm

2.1 Measurement and Error Model

In this paper, discretely sampled signals are studied. These could e.g. be audio or radio signals. Here, the sampling rate is assumed to be known and constant. Furthermore, we assume that the measured signal y has been ideally sampled after which noise – e.g. from the receivers – has been added, s.t.

$$y(k) = Y(k) + e(k)\,. \tag{1}$$

The original, continuous signal is denoted $Y : \mathbb{R} \mapsto \mathbb{R}$ and the noise, which is a discrete stationary stochastic process, is denoted e.

Let the set of functions $Y : \mathbb{R} \to \mathbb{R}$ that are (i) continuous (ii) square integrable and (iii) with a Fourier transform equal to zero outside $[-\pi, \pi]$ be denoted $\mathbb{B}$. Furthermore, denote the set of discrete, square integrable functions $y : \mathbb{Z} \to \mathbb{R}$ by ℓ. Now, define the discretization operator $D : \mathbb{B} \to \ell$ by

$$y(i) = D(Y)(i) = Y(i). \tag{2}$$

Moreover, we introduce the interpolation operator $I_g : \ell \to \mathbb{B}$, as

$$Y(x) = I_g(y)(x) = \sum_{i=-\infty}^{\infty} g(x-i)y(i)\,. \tag{3}$$

It has been shown that interpolation using the normalized sinc function, i.e. with $g(x) = \mathrm{sinc}(x)$, restores a sampled function for functions in $\mathbb{B}$, see [20] Thus, we call $I_{\mathrm{sinc}} : \ell \to \mathbb{B}$ the ideal interpolation operator and we have that

$$I_{\mathrm{sinc}}(D(Y)) = Y. \tag{4}$$

In the same way other interpolation methods can be expressed similarly. E.g. we obtain Gaussian interpolation by changing sinc in the expression above to

$$G_a(x) = \frac{1}{\sqrt{2\pi a^2}} \mathrm{e}^{x^2/(2a^2)}. \tag{5}$$

2.2 Scale-space Smoothing and Ideal Interpolation

A measured and interpolated signal is often smoothed for two reasons. Firstly, there is often more signal as compared to noise for lower frequencies, whereas for higher frequencies there is usually less signal in relation to noise. Therefore smoothing can be used in order to remove some of the noise, while keeping most of the signal.

Secondly, patterns in a more coarse scale are easier captured after smoothing has been applied, [15]. A Gaussian kernel G_{a_2}, with standard deviation a_2, has been used for the smoothing. We will also refer to a_2 as the *smoothing parameter*.

Given a sampled signal y, the ideally interpolated and smoothed signal can be written as

$$Y(x) = (G_{a_2} * I_{\mathrm{sinc}}(y))(x) = I_{G_{a_2} * \mathrm{sinc}}(y)(x). \tag{6}$$

If a_2 is large enough the approximation $G_{a_2} * \mathrm{sinc} \approx G_{a_2}$ holds. Thus, one can use interpolation with the Gaussian kernel as an approximation for ideal interpolation followed by Gaussian smoothing, [3], s.t.

$$Y(x) = I_{G_{a_2} * \mathrm{sinc}}(y)(x) \approx I_{G_{a_2}}(y)(x). \tag{7}$$

What *large enough* means will be studied in Section 4.1.

Moreover, we will later use the fact that discrete w.s.s. Gaussian noise interpolates to continuous w.s.s. Gaussian noise, as is shown in [3].

3 Time-difference and Doppler Estimation

Assume that we have two signals, $W(t)$ and $\bar{W}(t)$. The signals are measured and interpolated as described above. Also assume that the two signals are similar, but with one e,g. translated and compressed in the time domain. This could occur when two different receivers pick up an audio signal from a single sender. Then the second signal can be obtained from the other and a few parameters. We describe the relation as follows

$$W(t) = \bar{W}(\alpha t + h), \tag{8}$$

where h describes the time-difference of arrival, or translation in the signals. In a setup where the sound source has equal distance to both microphones $h = 0$. The second parameter, α, is a Doppler factor. This parameter is needed for example if the sound source or the microphones are moving. For a stationary setup $\alpha = 1$.

When the two microphones pick up the signals these are disturbed by Gaussian w.s.s. noise. Thus, the received signals can be written

$$V(t) = W(t) + E(t) \quad \text{and} \quad \bar{V}(t) = \bar{W}(t) + \bar{E}(t)\,. \tag{9}$$

Here, $E(t)$ and $\bar{E}(t)$ denotes the two independent noise signals after interpolation.

Assume that the signals V and $\bar{V}$ are given. Also, denote by $\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^T = \begin{bmatrix} h & \alpha \end{bmatrix}^T$, the vector of unknown parameters. Then, the parameters for which (8) is true can be estimated by the $\boldsymbol{z}$ that minimizes the integral

$$G(\boldsymbol{z}) = \int_t (V(t) - \bar{V}(z_2 t + z_1))^2 \,\mathrm{d}t. \tag{10}$$

Comparing with Cross Correlation

If we only estimate a time delay h, the minimization of the error function (10) would in practice be the same as maximizing the cross correlation of V and $\bar{V}$. The cross-correlation

for real signals is defined as

$$(V \star \bar{V})(h) = \int_t V(t)\bar{V}(t+h)\,\mathrm{d}t. \tag{11}$$

Thus, the h that maximize this cross-correlation is given by

$$\operatorname{argmax}_h (V \star \bar{V})(h) = \operatorname{argmax}_h \int_t V(t)\bar{V}(t+h)\,\mathrm{d}t. \tag{12}$$

If we expand the error function (10), while neglecting the Doppler factor we obtain the minimizer

$$\begin{aligned}\operatorname{argmin}_h \int_t (V(t) - \bar{V}(t+h))^2\,\mathrm{d}t &= \operatorname{argmin}_h \int_t (V(t))^2 + (\bar{V}(t+h))^2 - 2V(t)\bar{V}(t+h)\,\mathrm{d}t \\ &= \operatorname{argmin}_h \int_t -2V(t)\bar{V}(t+h)\,\mathrm{d}t = \operatorname{argmax}_h \int_t V(t)\bar{V}(t+h)\,\mathrm{d}t.\end{aligned} \tag{13}$$

Note that since we integrate over t, the integral $\int_t (\bar{V}(t+h))^2\,\mathrm{d}t$ is almost constant, ignoring edge effects.

We choose to use (10) for estimation of the parameters since it is simple to expand and is valid even if we add more parameters.

3.1 Estimating the Standard Deviation of the Parameters

If $\boldsymbol{z}_T = \begin{bmatrix} h_T & \alpha_T \end{bmatrix}^T$ is the "true" parameter for the data and $\hat{\boldsymbol{z}}$ is the parameter that has been estimated by minimizing (10), the estimation error can be expressed as

$$X = \hat{\boldsymbol{z}} - \boldsymbol{z}_T. \tag{14}$$

Assume, without loss of generality, that $\boldsymbol{z}_T = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$. The standard deviation of $\hat{\boldsymbol{z}}$ will be the same as the standard deviation of X and the mean of those two will only differ by $\boldsymbol{z}_T$. Thus, it is sufficient to study X to get statistical information about the estimate $\hat{\boldsymbol{z}}$.

Linearizing $G(\boldsymbol{z})$ around the true displacement $\boldsymbol{z}_T = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ gives

$$G(\boldsymbol{z}) \approx F(X) = \frac{1}{2}X^T aX + bX + f, \tag{15}$$

with

$$a = \nabla^2 G(\boldsymbol{z}_T)\,, \qquad b = \nabla G(\boldsymbol{z}_T)\,, \qquad f = G(\boldsymbol{z}_T)\,. \tag{16}$$

Using (9) and (8), we get

$$\begin{aligned} f = G(\begin{bmatrix} 0 & 1 \end{bmatrix}^T) &= \int_t (V(t) - \bar{V}(1 \cdot t - 0))^2 \,\mathrm{d}t = \int_t (W(t) + E(t) - (\bar{W}(t) + \bar{E}(t)))^2 \,\mathrm{d}t \\ &= \int_t (E - \bar{E})^2 \,\mathrm{d}t = \int_t E^2 + 2E\bar{E} + \bar{E}^2 \,\mathrm{d}t. \end{aligned} \tag{17}$$

To find the coefficients a and b we first calculate the derivatives $\nabla G(\boldsymbol{z})$ and $\nabla^2 G(\boldsymbol{z})$.

$$\begin{aligned} \nabla G &= \begin{bmatrix} \int_t 2(V(t) - \bar{V}(\alpha t + h)) \cdot (-\bar{V}'(\alpha t + h)) \,\mathrm{d}t \\ \int_t 2(V(t) - \bar{V}(\alpha t + h)) \cdot (-\bar{V}'(\alpha t + h) \cdot t) \,\mathrm{d}t \end{bmatrix} \\ &= -2 \begin{bmatrix} \int_t (W(t) + E(t) - \bar{W}(\alpha t + h) - \bar{E}(\alpha t + h))(\bar{W}'(\alpha t + h) + \bar{E}'(\alpha t + h)) \,\mathrm{d}t \\ \int_t (W(t) + E(t) - \bar{W}(\alpha t + h) - \bar{E}(\alpha t + h))(\bar{W}'(\alpha t + h) + \bar{E}'(\alpha t + h)) \cdot t \,\mathrm{d}t \end{bmatrix}. \end{aligned} \tag{18}$$

Inserting the true displacement $\boldsymbol{z}_T$, at the point of linearization, gives

$$\begin{aligned} b &= \nabla G(\boldsymbol{z}_T) \\ &= -2 \begin{bmatrix} \int_t (W(t) + E(t) - \bar{W}(t) - \bar{E}(t)) \cdot (\bar{W}'(t) + \bar{E}'(t)) \,\mathrm{d}t \\ \int_t (W(t) + E(t) - \bar{W}(t) - \bar{E}(t)) \cdot (\bar{W}'(t) + \bar{E}'(t)) \cdot t \,\mathrm{d}t \end{bmatrix} \\ &= -2 \begin{bmatrix} \int_t (E - \bar{E})(\bar{W}' + \bar{E}') \,\mathrm{d}t \\ \int_t (E - \bar{E})(\bar{W}' + \bar{E}') t \,\mathrm{d}t \end{bmatrix} = -2 \begin{bmatrix} \int_t E\bar{W}' + E\bar{E}' - \bar{E}\bar{W}' - \bar{E}\bar{E}' \,\mathrm{d}t \\ \int_t (E\bar{W}' + E\bar{E}' - \bar{E}\bar{W}' - \bar{E}\bar{E}') t \,\mathrm{d}t \end{bmatrix}. \end{aligned} \tag{19}$$

To simplify further computations, we introduce

$$\hat{\varphi} = E\bar{W}' + E\bar{E}' - \bar{E}\bar{W}' - \bar{E}\bar{E}', \tag{20}$$

such that

$$b = -2 \begin{bmatrix} \int_t \hat{\varphi} \,\mathrm{d}t \\ \int_t t\hat{\varphi} \,\mathrm{d}t \end{bmatrix}. \tag{21}$$

Furthermore,

$$\begin{aligned} &\nabla^2 G = \\ &\begin{bmatrix} \int_t -2\bar{V}'(\alpha t + h) \cdot (-\bar{V}'(\alpha t + h)) + 2(V(t) - \bar{V}(\alpha t + h))(-\bar{V}''(\alpha t + h)) \,\mathrm{d}t & \cdots \\ \int_t -2\bar{V}'(\alpha t + h) \cdot t \cdot (-\bar{V}'(\alpha t + h)) + 2(V(t) - \bar{V}(\alpha t + h))(-\bar{V}''(\alpha t + h) \cdot t) \,\mathrm{d}t & \cdots \\ \int_t -2\bar{V}'(\alpha t + h) \cdot (-\bar{V}'(\alpha t + h)) \cdot t + 2(V(t) - \bar{V}(\alpha t + h)) \cdot (-\bar{V}''(\alpha t + h) \cdot t) \,\mathrm{d}t & \\ \int_t -2\bar{V}'(\alpha t + h) \cdot t(-\bar{V}'(\alpha t + h)) \cdot t + 2(V(t) - \bar{V}(\alpha t + h)) \cdot (-\bar{V}''(\alpha t + h) \cdot t^2) \,\mathrm{d}t & \end{bmatrix} \\ &= 2 \begin{bmatrix} \int_t (\bar{V}'(\alpha t + h))^2 - V(t)\bar{V}''(\alpha t + h) + \bar{V}(\alpha t + h)\bar{V}''(\alpha t + h) \,\mathrm{d}t & \cdots \\ \int_t t \cdot (\bar{V}'(\alpha t + h))^2 - t \cdot V(t)\bar{V}''(\alpha t + h) + t \cdot \bar{V}(\alpha t + h)\bar{V}''(\alpha t + h) \,\mathrm{d}t & \cdots \\ \int_t \cdot (\bar{V}'(\alpha t + h))^2 - t \cdot V(t)\bar{V}''(\alpha t + h) + t \cdot \bar{V}(\alpha t + h)\bar{V}''(\alpha t + h) \,\mathrm{d}t & \\ \int_t t^2 \cdot (\bar{V}'(\alpha t + h))^2 - t^2 \cdot V(t)\bar{V}''(\alpha t + h) + t^2 \cdot \bar{V}(\alpha t + h)\bar{V}''(\alpha t + h) \,\mathrm{d}t & \end{bmatrix}. \end{aligned} \tag{22}$$

Now, introducing the notation

$$\phi(\boldsymbol{z}) = (\bar{V}'(\alpha t+h))^2 - V(t)\bar{V}''(\alpha t+h) + \bar{V}(\alpha t+h)\bar{V}''(\alpha t+h), \tag{23}$$

we can write $\nabla^2 G$ shorter as

$$\nabla^2 G = 2\begin{bmatrix} \int_t \phi \,\mathrm{d}t & \int_t t\phi \,\mathrm{d}t \\ \int_t t\phi \,\mathrm{d}t & \int_t t^2\phi \,\mathrm{d}t \end{bmatrix}. \tag{24}$$

If we let $\hat{\phi}$ be the value of ϕ for $\boldsymbol{z}_T$

$$\begin{aligned} \hat{\phi} =&\phi(\boldsymbol{z}_T) = (\bar{W}'(t) + \bar{E}'(t))^2 - (W(t) + E(t))(\bar{W}''(t) + \bar{E}''(t)) \\ &+ (\bar{W}(t) + \bar{E}(t))(\bar{W}''(t) + \bar{E}''(t)) \\ =&(\bar{W}')^2 + 2\bar{W}'\bar{E}' + (\bar{E}')^2 - E\bar{W}'' - E\bar{E}'' + \bar{E}\bar{W}'' + \bar{E}\bar{E}'' \end{aligned} \tag{25}$$

we get

$$a = \nabla^2 G(\boldsymbol{z}_T) = 2\begin{bmatrix} \int_t \hat{\phi} \,\mathrm{d}t & \int_t t\hat{\phi} \,\mathrm{d}t \\ \int_t t\hat{\phi} \,\mathrm{d}t & \int_t t^2\hat{\phi} \,\mathrm{d}t \end{bmatrix}. \tag{26}$$

We also have that $F(X) = 1/2 \cdot X^T aX + bX + f$. To minimize this error function, we find the X for which the derivative of $F(X)$ is zero. Since a is symmetric we get

$$\nabla F(X) = aX + b = 0 \quad \Leftrightarrow \quad X = g(a,b) = -a^{-1}b. \tag{27}$$

In the calculations below, we assume that a is invertible.

Now we would like to find the mean and covariance of X. For this, Gauss' approximation formulas are used. If we denote the expected value of a and b with $\mu_a = \mathbf{E}[A]$ and $\mu_b = \mathbf{E}[b]$ respectively the expected value of X can be approximated to

$$\begin{aligned} \mathbf{E}[X] =&\mathbf{E}[g(a,b)] \approx \mathbf{E}[g(\mu_a,\mu_b) + (a-\mu_a)g'_a(\mu_a,\mu_b) + (b-\mu_b)g'_b(\mu_a,\mu_b)] \\ =&g(\mu_a,\mu_b) + (\mathbf{E}[a]-\mu_a)g'_a(\mu_a,\mu_b) + (\mathbf{E}[b]-\mu_b)g'_b(\mu_a,\mu_b) \\ =&g(\mu_a,\mu_b) = -\mu_a^{-1}\mu_b = -\mathbf{E}[a]^{-1}\mathbf{E}[b]. \end{aligned} \tag{28}$$

In a similar manner the covariance of X is

$$\begin{aligned} \mathbf{C}[X] =&\mathbf{C}[g(a,b)] \approx g'_a(\mu_a,\mu_b)\mathbf{C}[a]g'_a(\mu_a,\mu_b)^T + g'_b(\mu_a,\mu_b)\mathbf{C}[b]g'_b(\mu_a,\mu_b)^T \\ &+ 2g'_a(\mu_a,\mu_b)\mathbf{C}[a,b]g'_b(\mu_a,\mu_b)^T, \end{aligned} \tag{29}$$

where $\mathbf{C}[a,b]$ denotes the cross-covariance between a and b. For further computations $g'_a(a,b)$, $g'_b(a,b)$, $\mathbf{E}[a]$, $\mathbf{E}[b]$, $\mathbf{C}[b]$ and $\mathbf{C}[a,b]$ are needed.

By computing the expected value of $\hat{\varphi}$

$$\begin{aligned}\mathbf{E}[\hat{\varphi}] =&\mathbf{E}[E\bar{W}' + E\bar{E}' - \bar{E}\bar{W}' - \bar{E}\bar{E}'] \\ =&\mathbf{E}[E]\,\bar{W}' + \mathbf{E}[E]\mathbf{E}[\bar{E}'] - \mathbf{E}[\bar{E}]\,\bar{W}' - \mathbf{E}[\bar{E}]\mathbf{E}[\bar{E}'] = 0\end{aligned} \tag{30}$$

we get

$$\mathbf{E}[b] =\mathbf{E}\left[-2\begin{bmatrix}\int_t \hat{\varphi}\ dt \\ \int_t t\hat{\varphi}\,\mathrm{d}t\end{bmatrix}\right] = -2\begin{bmatrix}\int_t \mathbf{E}[\hat{\varphi}]\,\mathrm{d}t \\ \int_t t\mathbf{E}[\hat{\varphi}]\,\mathrm{d}t\end{bmatrix} = -2\begin{bmatrix}\int_t 0\,\mathrm{d}t \\ \int_t t\cdot 0\,\mathrm{d}t\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}. \tag{31}$$

In the second step of the computation of $\mathbf{E}[\hat{\varphi}]$ we have used the fact that for a weakly stationary process the process and its derivative at a certain time are uncorrelated, and thus $\mathbf{E}[\bar{E}\bar{E}'] = \mathbf{E}[\bar{E}]\mathbf{E}[\bar{E}']$, [16]. Hence,

$$\mathbf{E}[X] = -\mathbf{E}[a]^{-1}\mathbf{E}[b] = -\mathbf{E}[a]^{-1}\begin{bmatrix}0\\0\end{bmatrix} = \begin{bmatrix}0\\0\end{bmatrix}. \tag{32}$$

For the partial derivative of $g(a,b)$ w.r.t. b we get [17]

$$g_b'(a,b) = \frac{\partial}{\partial b}\left(-a^{-1}b\right) = -(a^{-1})^T = -(a^T)^{-1} = -a^{-1} \tag{33}$$

and thus $g_b'(\mu_a,\mu_b) = -(\mathbf{E}[a])^{-1}$. Since $\mathbf{E}[b] = \mathbf{0}$, we get that $g_a'(\mu_a,\mu_b) = \mathbf{0}$, [17]. Hence the first and the last term in (29) cancel, leaving

$$\begin{aligned}\mathbf{C}[X] =&g_b'(\mu_a,\mu_b)\mathbf{C}[b]g_b'(\mu_a,\mu_b)^T = (-\mathbf{E}[a]^{-1})\mathbf{C}[b](-\mathbf{E}[a]^{-1})^T \\ =&\mathbf{E}[a]^{-1}\mathbf{C}[b](\mathbf{E}[a]^{-1})^T.\end{aligned} \tag{34}$$

To find the expected value of a the expected value of $\hat{\phi}$ is needed. This is obtained from

$$\begin{aligned}\mathbf{E}[\hat{\phi}] =&(\bar{W}')^2 + 2\bar{W}'\mathbf{E}[\bar{E}'] + \mathbf{E}[(\bar{E}')^2] - \bar{W}''\mathbf{E}[E] - \mathbf{E}[E]\mathbf{E}[\bar{E}''] + \bar{W}''\mathbf{E}[\bar{E}] + \mathbf{E}[\bar{E}\bar{E}''] \\ =&(\bar{W}')^2 + \mathbf{E}[(\bar{E}')^2] + \mathbf{E}[\bar{E}\bar{E}''] = (\bar{W}')^2.\end{aligned} \tag{35}$$

In the last equality we have used that $\mathbf{E}[\bar{E}\bar{E}''] = -\mathbf{E}[(\bar{E}')^2]$, [16]. Thus, the two last terms cancel out. The expected value of a is therefore

$$\mathbf{E}[a] =2\begin{bmatrix}\int_t \mathbf{E}[\hat{\phi}]\,\mathrm{d}t & \int_t \mathbf{E}[t\hat{\phi}]\,\mathrm{d}t \\ \int_t \mathbf{E}[t\hat{\phi}]\,\mathrm{d}t & \int_t \mathbf{E}[t^2\hat{\phi}]\,\mathrm{d}t\end{bmatrix} = 2\begin{bmatrix}\int_t (\bar{W}')^2\,\mathrm{d}t & \int_t t(\bar{W}')^2\,\mathrm{d}t \\ \int_t t(\bar{W}')^2\,\mathrm{d}t & \int_t t^2(\bar{W}')^2\,\mathrm{d}t\end{bmatrix}. \tag{36}$$

Now, since the expected value of b is zero, the covariance of b is

$$\mathbf{C}[b] = (-2)^2\begin{bmatrix}C_{11} & C_{12} \\ C_{21} & C_{22}\end{bmatrix}, \tag{37}$$

with

$$
\begin{aligned}
C_{11} &= \mathbf{E}\left[\int_{t_1} \hat{\varphi}(t_1)\,\mathrm{d}t_1 \cdot \int_{t_2} \hat{\varphi}(t_2)\,\mathrm{d}t_2\right] \\
C_{12} &= \mathbf{E}\left[\int_{t_1} t_1\hat{\varphi}(t_1)\,\mathrm{d}t_1 \cdot \int_{t_2} \hat{\varphi}(t_2)\,\mathrm{d}t_2\right] \\
C_{21} &= \mathbf{E}\left[\int_{t_1} \hat{\varphi}(t_1)\,\mathrm{d}t_1 \cdot \int_{t_2} t_2\hat{\varphi}(t_2)\,\mathrm{d}t_2\right] \\
C_{22} &= \mathbf{E}\left[\int_{t_1} t_1\hat{\varphi}(t_1)\,\mathrm{d}t_1 \cdot \int_{t_2} t_2\hat{\varphi}(t_2)\,\mathrm{d}t_2\right] .
\end{aligned}
\tag{38}
$$

Note that by changing the order of the terms in C_{12} it is clear that $C_{21} = C_{12}$. Furthermore, we obtain

$$
\begin{aligned}
C_{11} =& \mathbf{E}\left[\int_{t_1} \hat{\varphi}(t_1)\,\mathrm{d}t_1 \cdot \int_{t_2} \hat{\varphi}(t_2)\,\mathrm{d}t_2\right] \\
=& \mathbf{E}\Big[\Big(\int_{t_1} (E-\bar{E})(\bar{W}'+\bar{E}')\,\mathrm{d}t_1\Big) \cdot \Big(\int_{t_2} (E-\bar{E})(\bar{W}'+\bar{E}')\,\mathrm{d}t_2\Big)\Big] \\
=& \mathbf{E}\Big[\int_{t_1}\int_{t_2} (E(t_1)-\bar{E}(t_1))(\bar{W}'(t_1)+\bar{E}'(t_1)) \\
& \cdot (E(t_2)-\bar{E}(t_2))(\bar{W}'(t_2)+\bar{E}'(t_2))\,\mathrm{d}t_1\mathrm{d}t_2\Big] .
\end{aligned}
\tag{39}
$$

Denoting $\mathbf{E}[(E(t_1)-\bar{E}(t_1))(E(t_2)-\bar{E}(t_2))] = r_{E-\bar{E}}(t_1-t_2)$ and assuming that $\mathbf{E}[\bar{E}'(t_1)\bar{E}'(t_2)]$ is small gives

$$
\begin{aligned}
C_{11} =& \mathbf{E}\left[\int_{t_1} \hat{\varphi}(t_1)\,\mathrm{d}t_1 \cdot \int_{t_2} \hat{\varphi}(t_2)\,\mathrm{d}t_2\right] \\
=& \int_{t_1}\int_{t_2} \mathbf{E}[(E(t_1)-\bar{E}(t_1))(E(t_2)-\bar{E}(t_2))] \cdot (\bar{W}'(t_1)\bar{W}'(t_2) \\
& + \bar{W}'(t_1)\mathbf{E}[\bar{E}'(t_2)] + \mathbf{E}[\bar{E}'(t_1)]\bar{W}'(t_2) + \mathbf{E}[\bar{E}'(t_1)\bar{E}'(t_2)])\,\mathrm{d}t_2\mathrm{d}t_1 \\
\approx& \int_{t_1}\int_{t_2} r_{E-\bar{E}}(t_1-t_2)\bar{W}'(t_1)\bar{W}'(t_2)\,\mathrm{d}t_2\mathrm{d}t_1 \\
=& \int_{t_1} \bar{W}'(t_1)(\bar{W}' * r_{E-\bar{E}})(t_1)\,\mathrm{d}t_1 .
\end{aligned}
\tag{40}
$$

The time t is a deterministic quantity and the other elements in $\mathbf{C}[b]$ can be computed

similarly. Finally we have

$$\begin{aligned} C_{11} &= \int_t \bar{W}'(t)(\bar{W}' * r_{E-\bar{E}})(t)\,\mathrm{d}t \\ C_{12} = C_{21} &= \int_t t\bar{W}'(t)(\bar{W}' * r_{E-\bar{E}})(t)\,\mathrm{d}t \\ C_{22} &= \int_t t\bar{W}'(t)((t\bar{W}') * r_{E-\bar{E}})(t)\,\mathrm{d}t \end{aligned} \tag{41}$$

and through (34) we get an expression for the variance and thus also the standard deviation of X.

3.2 Expanding the Model

It is easy to change or expand the model (8) to contain more (or fewer) parameters. If we keep h and α and add an extra amplitude parameter γ, we get the model

$$W(t) = \gamma\bar{W}(\alpha t + h). \tag{42}$$

The error integral (10) would then be changed accordingly and the optimization would instead be over over $\boldsymbol{z} = \begin{bmatrix} z_1 & z_2 & z_3 \end{bmatrix} = \begin{bmatrix} h & \alpha & \gamma \end{bmatrix}$.

The computations for achieving the estimations does in practice not get harder when we add more parameters. However, the analysis from the previous section gets more complex.

4 Experimental Validation

For validation we perform experiments on both real data and synthetic data. The purpose of using synthetic data is to demonstrate the validity of the model, but also to verify the approximations used. In the latter case we have studied at what signal-to-noise ratio the approximations are valid. Furthermore, to show that the parameter estimations contain useful information, we have done experiments on real data. This is well-known for time-difference, but less explored for the Doppler effects and amplitude changes.

4.1 Synthetic data - Validation of Method

The model was first tested on simulated data in order to study when the approximations in the model derivation hold. The linearization using Gauss' approximation formula, e.g. (28) and (29), is one example of such approximations. Another is the usage of Gaussian

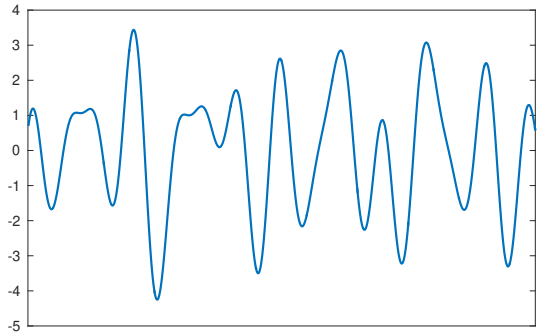


Figure 3: The simulated signal that was used for the experimental validation. To achieve a more realistic signal noise of different levels was added later on. The plot is taken from [10].

interpolation as an approximation of ideal interpolation followed by convolution with a Gaussian, (7).

To do these studies we compared the theoretical standard deviations of the parameters calculated according to Section 3.1 with empirically computed standard deviations. The agreement of these standard deviations makes us conclude that our approximations are valid.

First we simulated an original continuous signal $W(x)$, see Figure 3. The second signal was then created according to (8) s.t. $\bar{W}(x) = W(1/\alpha \cdot (x - h))$. The signals were ideally sampled after which Gaussian white discrete noise with standard deviation σ_n was added. After smoothing with a Gaussian kernel with standard deviation a_2 (see Section 2.2) the signals can be described by $V(t)$ and $\bar{V}(t)$ as before.

The two signals V and $\bar{V}$ were simulated anew 1000 times to investigate the effect of a_2 and σ_n. Each time the same original signals W and $\bar{W}$ were used, but with different noise realizations. Then, we computed the theoretical standard deviation of the parameter vector $\boldsymbol{z}$, $\sigma_{\boldsymbol{z}} = \begin{bmatrix}\sigma_h & \sigma_\alpha\end{bmatrix}$. This was done in accordance with the presented theory. We also computed an empirical standard deviation $\hat{\sigma}_{\boldsymbol{z}} = \begin{bmatrix}\hat{\sigma}_h & \hat{\sigma}_\alpha\end{bmatrix}$ from the 1000 different parameter estimations.

When studying the effect of a_2 the noise level was kept constant, with $\sigma_n = 0.03$. The translation was set to $h = 3.63$ and the Doppler factor was $\alpha = 1.02$. However, the exact numbers are unessential. While varying the smoothing parameter $a_2 \in [0.3, 0.8]$ the standard deviation was then computed according to the procedure above.

The results from these simulations can be seen in Figure 4. When a_2 is below $a_2 \approx 0.55$ the theoretical values $\sigma_{\boldsymbol{z}}$ and the empirical values $\hat{\sigma}_{\boldsymbol{z}}$ do not agree, while they do for $a_2 > 0.55$.

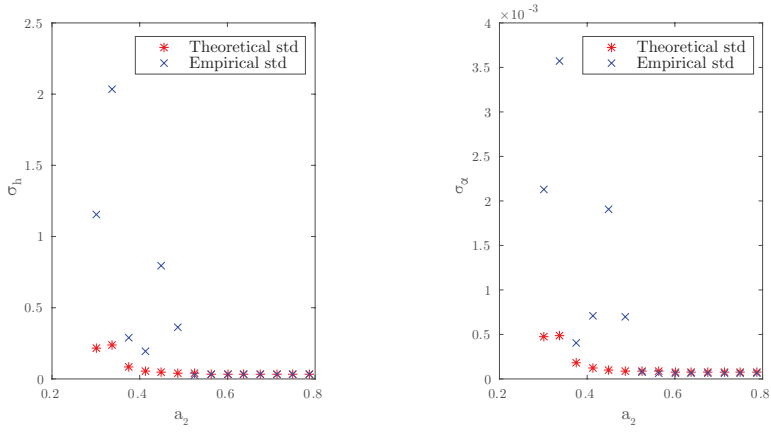


Figure 4: The plots show the standard deviation of the parameters in $\boldsymbol{z}$ for different values of the smoothing parameter a_2. The stars ($*$) represent the theoretical values $\sigma_{\boldsymbol{z}}$ and the crosses (x) the empirical values $\hat{\sigma}_{\boldsymbol{z}}$. The left plot shows the results for the translation $z_1 = h$ and the right plot for the Doppler factor $z_2 = \alpha$. It is clear that the approximation is valid approximately when $a_2 > 0.55$. The plots are taken from [10].

Therefore we draw the conclusion that the approximation (7) of ideal interpolation should only be used when $a_2 > 0.55$.

Secondly, the effect of changing the noise level was investigated. The smoothing parameters was fixed to $a_2 = 2$ and the translation and the Doppler factor were kept on the same level as before. Instead we varied the noise level s.t. $\sigma_n \in [0, 1.6]$. Then the standard deviations of the parameters $\sigma_{\boldsymbol{z}}$ and $\hat{\sigma}_{\boldsymbol{z}}$ were computed in the same way as in the previous section.

The results from this run can be seen in Figure 5, with the results for the translation parameter h to the left and for the Doppler parameter α to the right. When σ_n is lower than $\sigma_n \approx 0.8$ the theoretical and empirical values for the translation parameter are similar. For higher values of σ_n they do not agree. The same goes for the Doppler factor when the noise level is below $\sigma_n \approx 1.1$.

By this, we reason that noise with a standard deviation up to $\sigma_n \approx 0.8$ can be handled. The original signal W have an amplitude that varies between 1 and 3.5 and using the standard deviation of that signal, σ_W, we can compute the signal-to-noise ratio that the system can manage. We get the result

$$\text{SNR} = \frac{\sigma_W^2}{\sigma_n^2} \approx 4.7. \tag{43}$$

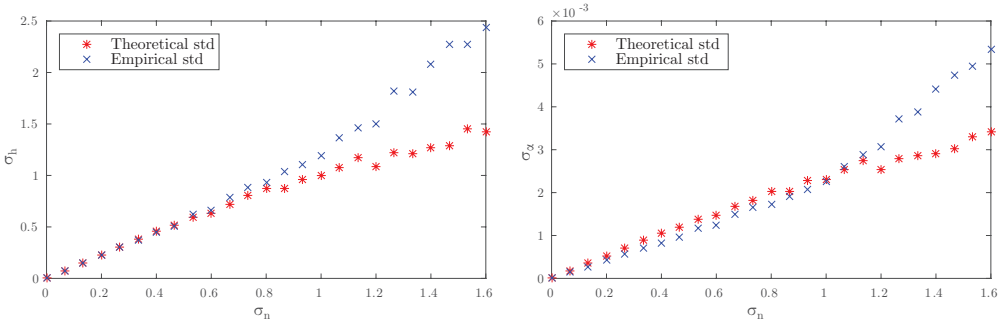


Figure 5: The standard deviation of the translation (to the left) and Doppler factor (to the right) for different levels of noise in the signal. The stars (∗) mark the theoretical values $\sigma_{\boldsymbol{z}}$ and the crosses (x) the empirical $\hat{\sigma}_{\boldsymbol{z}}$. For the translation the values agree for signals with a noise level up to $\sigma_n \approx 0.8$. For the Doppler factor the theoretical values follow the empirical values when $\sigma_n < 1.1$. The plots are taken from [10].

Comparing Different Models

In this paper we have chosen to work with the models (8) and (42). However, we have so far not presented any comparison between different models. To investigate this, we studied two models, namely (8), which we call model B and a slightly simpler model which we call model A,

$$W(x) = \bar{W}(x + h). \tag{44}$$

To begin with, we simulated data according to model A. We call this data A. During the simulation the standard deviation of the noise in the signals was set to $\sigma_n = 0.02$ and the smoothing parameter was $a_2 = 2.0$. Furthermore, we studied this data both using model A, i.e. by minimizing $\int_t (V(t) - \bar{V}(t+h))^2 \, \mathrm{d}t$ and using model B, see (10). The results can be seen in the first column (Data A) of Table 1.

Secondly, a similar test was made but this time we simulated data according to model B. We call this data B. We then studied this data using both model A and B. The results are shown in the second column (data B) of Table 1.

Studying the first column of Table 1 we see that model B estimates the parameters as good as model A – which in this case is the most correct model – does. Though, for model B the standard deviation σ_h is more than twice as big as for model A.

In the second column of the table we see that since model A cannot estimate the Doppler effect, the translation parameter is erroneously estimated. The standard deviation σ_h is however still lower for model A. To minimize the error function model A estimates the translation such that the signal is fitted in the middle, see Figure 6. This means that even though the standard deviation is low, the bias is high.

If we know that our collected data has only been affected by a translation it is clearly better

Table 1: Comparison between model A from (44) and model B from (8). Data A consists of signals with only translational differences while the second signal in data B is affected by both translation and a Doppler effect. The standard deviations for model B in the table regards the theoretical values that were derived in Section 3.1, and a similar analysis has been performed for model A.

		Data A	**Data B**
True values	Translation, h_T	3.63	3.63
	Doppler factor, α_T	1.00	1.02
Model A	Est. h, $\hat{h}^{(A)}$	3.63	13.4
	Std. of h, $\sigma_h^{(A)}$	$1.01 \cdot 10^{-2}$	$1.02 \cdot 10^{-2}$
Model B	Est. h, $\hat{h}^{(B)}$	3.63	3.66
	Std. of h, $\sigma_h^{(B)}$	$2.30 \cdot 10^{-2}$	$2.30 \cdot 10^{-2}$
	Est. α, $\hat{\alpha}^{(B)}$	1.00	1.02
	Std. of α, $\sigma_\alpha^{(B)}$	$5.32 \cdot 10^{-5}$	$5.33 \cdot 10^{-5}$

to use model A. However, the loss for using a more simple model is larger on complex data than the loss for using a larger model for simple data. Thus, based on the results from Table 1 we conclude that it is better to use a larger model for the real data in the following section.

4.2 Real data - Validation of Method

The experiments on real data were performed in an anaechoic chamber and the recording frequency was $f = 96$ kHz. We used 8 T-Bone MM-1 microphones and these were connected to an audio interface (M-Audio Fast Track Ultra 8R) and a computer. Furthermore, the microphones were placed so that they spanned 3D, approximately 0.3-1.5 meters away from each other. As a sound source we used a mobile phone which was connected to a small loudspeaker. The mobile phone was moved around in the room while playing a song.

We used the technique described in [22] and refined in [21] to achieve ground truth consisting of a 3D trajectory for the sound source path $s(t)$ and the 3D positions of the microphones $r_1, \ldots, r_8$. The method uses RANSAC algorithms which are based on minimal solvers [14] to find initial estimates of the sound trajectory and microphone positions. Then, these are refined using non-linear optimization of a robust error norm, including a smooth motion prior, to reach the final estimates.

However, to make ground truth independent from the data that we used for testing we chose to only take data from microphone 3-8 into account during the first two thirds of the sound signal. Thus, by that we estimated $s(t)$ for certain t and $r_3, \ldots, r_8$. For the final third of the signal we added the information from microphone 1 and 2 as well, such that our solution would not drift compared to ground truth. By that we estimated the rest of $s(t)$, r_1 and r_2.

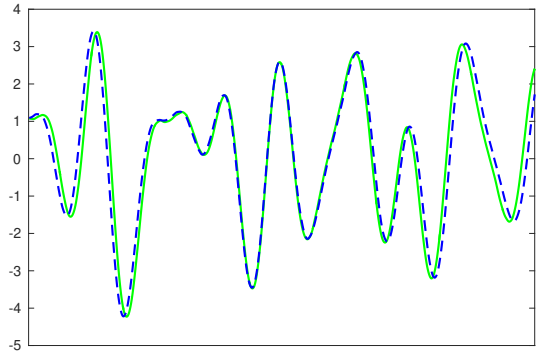


Figure 6: The results after using model A on data B, where the second signal is affected both by a translation and Doppler effect. Since the model does not estimate any Doppler factor, the estimated translation will be biased. The two signals agree well in the middle, while there is a gap between them at the beginning and the end. This gap cannot be captured by a translation.

We only used data from microphone 1 and 2 for the validation of the method presented in this paper. The sound was played for around 29 sec. and the loudspeaker was constantly moving during this time. Furthermore, both the direction and the speed of the sound source changed.

Since our method assume a constant parameter $\boldsymbol{z}$ in a window we divided the recording into 2834 patches of 1000 samples each (i.e. about 0.01 sec.). Within these patches the parameters were approximately constant. Each of the patches could then be investigated and compared to ground truth separately. From ground truth we had a constant loudspeaker position $s^{(i)}$, its derivative $\frac{\partial s^{(i)}}{\partial t}(i)$ and the receiver positions r_1 and r_2 for each signal patch i.

Estimating the Parameters

If we call signal patch i from the first microphone $V^{(i)}(t)$ and let $\bar{V}^{(i)}(t)$ be the patch from the second microphone we can estimate the parameters using (8) to model the received signals.

The method presented in this paper is developed to estimate small translations, s.t. $h \in [-10, 10]$ samples. However, in the experiments the delays were larger than that. Therefore we began by pre-estimating an integer delay $\tilde{h}^{(i)}$ using GCC-PHAT. The GCC-PHAT method is described in [11]. After that we did a subsample refinement of the translation and estimated the Doppler parameter using our method. This was done by minimization

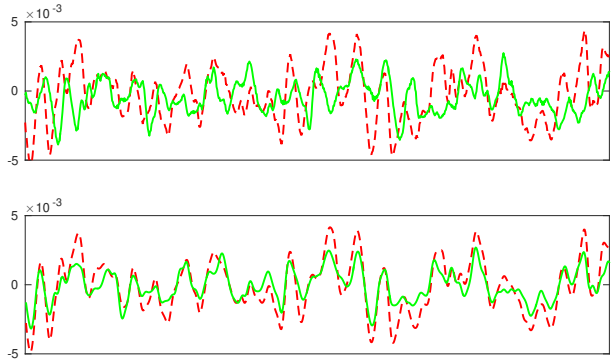


Figure 7: The received signal patches at a certain time – the first signal in dashed (- -) and the second as solid (—). The top plot shows the signals as they were received. In the lower plot the same patches have been modified using the optimal parameters h and α.

of the intergral

$$\int_t (V^{(i)}(t) - \bar{V}^{(i)}(\alpha^{(i)}t + \tilde{h}^{(i)} + h^{(i)}))^2 \, \mathrm{d}t. \tag{45}$$

Here, the optimization was over $h^{(i)}$ and $\alpha^{(i)}$, while $\tilde{h}^{(i)}$ should be seen as a constant.

The results after applying the optimized parameters to one of the signal patches can be seen in Figure 7. The optimization was carried out for all different patches.

Comparison with Ground Truth

The distances $d_1^{(i)}$ and $d_2^{(i)}$ from the microphones to the loudspeaker were computed from the ground truth receiver and sender positions (r_1, r_2 and $s^{(i)}$) according to

$$d_1^{(i)} = |r_1 - s^{(i)}|, \qquad d_2^{(i)} = |r_2 - s^{(i)}|. \tag{46}$$

The difference of these distances,

$$\Delta d^{(i)} = d_2^{(i)} - d_1^{(i)} \tag{47}$$

has a connection to our estimated translation $h^{(i)}$ and the time difference of arrival. However, $\Delta d^{(i)}$ is measured in meters, while we compute $h^{(i)}$ in samples. To be able to compare these two, we multiplied $h^{(i)}$ with a scaling factor c/f. The recording frequency was $f = 96$ kHz and $c = 340$ m/s is the speed of sound. From this we could obtain an estimation of $\Delta d^{(i)}$,

$$\Delta \bar{d}^{(i)} = \frac{c}{f} \cdot h^{(i)}. \tag{48}$$

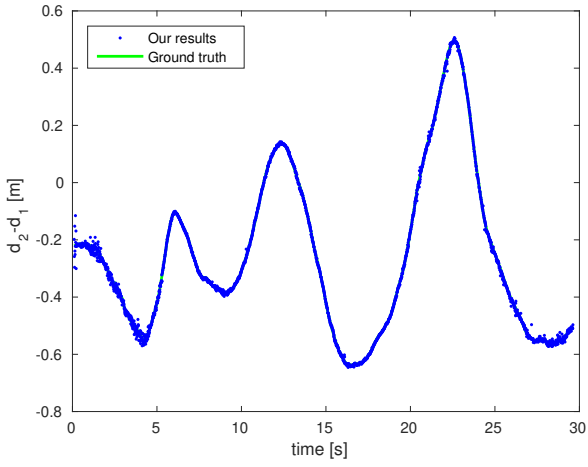


Figure 8: The figure shows the difference between the distances from receiver 1 to the sender (d_1) and receiver 2 to the sender (d_2) over time. The ground truth $\Delta d^{(i)}$ is plotted as a solid line (—) and the values $\Delta\bar{d}^{(i)}$ obtained from time-difference estimates as dots (•). Each dot represents the value for one signal patch. It is hard to distinguish the line representing ground truth since the estimations agree well with this. The plot is similar to Figure 7 in [10], but has been generated using the updated and more independent method which is presented in this paper.

Thereafter we could compare our estimated values $\Delta\bar{d}^{(i)}$ to the ground truth values $\Delta d^{(i)}$. The ground truth is plotted together with our estimations in Figure 8. The plot shows the results over time, for all different patches. It is clear that the two agree well.

The Doppler parameter measures how the distance differences changes, i.e.

$$\frac{\partial \Delta d}{\partial t} = \frac{\partial d_2}{\partial t} - \frac{\partial d_1}{\partial t}. \tag{49}$$

Here, the distances over time are denoted d_1 and d_2 respectively. The derivative of $d_1(t) = |r_1 - s(t)|$ is

$$\frac{\partial d_1}{\partial t} = \frac{r_1 - s}{|r_1 - s|} \cdot \frac{\partial s}{\partial t}, \tag{50}$$

where $\cdot$ denotes the scalar product between the two time dependent vectors. The derivative of d_2 can be found correspondingly. If $n_1^{(i)}$ and $n_2^{(i)}$ are unit vectors in the direction from $s^{(i)}$ to r_1 and r_2 respectively, i.e.

$$n_1^{(i)} = \frac{r_1 - s^{(i)}}{|r_1 - s^{(i)}|}, \qquad n_2^{(i)} = \frac{r_2 - s^{(i)}}{|r_2 - s^{(i)}|}, \tag{51}$$

the derivatives can be expressed as

$$\frac{\partial d_1^{(i)}}{\partial t} = n_1^{(i)} \cdot \frac{\partial s^{(i)}}{\partial t}, \qquad \frac{\partial d_2^{(i)}}{\partial t} = n_2^{(i)} \cdot \frac{\partial s^{(i)}}{\partial t}. \tag{52}$$

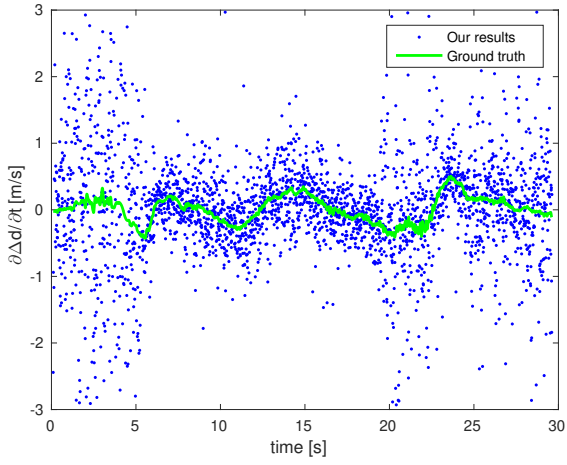


Figure 9: The derivative of the distance differences Δd plotted over time. The dots (•) are our estimations and the solid line (—) is computed from ground truth. We see that even though the estimations are noisy the pattern agree with ground truth. The plot is similar to Figure 8 in [10], but has been generated using the updated and more independent method which is presented in this paper.

Thus

$$\frac{\partial \Delta d^{(i)}}{\partial t} = n_2^{(i)} \cdot \frac{\partial s^{(i)}}{\partial t} - n_1^{(i)} \cdot \frac{\partial s^{(i)}}{\partial t}. \tag{53}$$

These ground truth Doppler values can be interpreted as how much Δd changes each second. However, our estimated Doppler factor α is a unit-less constant. We can express the relation between the two values as

$$\frac{\partial \Delta d}{\partial t} = (\alpha - 1) \cdot c, \tag{54}$$

where c still denotes the speed of sound. In Figure 9 the ground truth is plotted as a solid line together with our estimations marked with dots. The similarities are easily distinguishable even if the estimations are noisy.

It is clear from the plots that the estimations contain relevant information. However, there is quite some noise in the estimates in Figure 8 and 9. This can be reduced further by computation of a moving average. We have computed a moving average over 20 patches – approximately 0.2 sec. – for the distance difference derivative and plotted the result in Figure 10. The plot can be compared to Figure 9, where no averaging has been done. We see that the moving average substantially reduces the noise in the estimates.

Even in Figure 10, the estimates in the beginning are noisy. This is due to the character of the song that was played, where the sound is not persistent until after 5-6 sec. In the beginning there are just intermittent drumbeats and silence between these. Then the in-

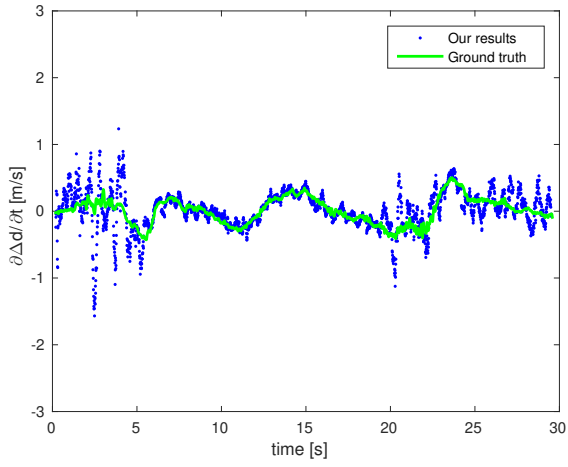


Figure 10: This plot shows essentially the same thing as Figure 9, i.e. $\partial \Delta d / \partial t$, but with a 20-patches moving average over the estimations. The averaging substantially reduces the noise. The plot is similar to Figure 10 in [10], but has been generated using the updated and more independent method which is presented in this paper.

formation is not sufficient to make good estimates. Thus, it is more fair to the algorithm to review the results from 5-6 sec. and forward.

Estimating the Standard Deviation of the Parameters

We have also computed the standard deviations of the parameters in accordance to Section 3.1. These are plotted over time in Figure 11. We can see that the estimations are more uncertain in the beginning of the song, in consistence with when the signal is not persistent. However, just by looking at the estimated Doppler factor this seems to be more uncertain than the theoretical standard deviation suggests.

We also estimated the standard deviations empirically. This was done using the results in Figure 8 and 9. The empirical standard deviation was computed for the difference between our estimations and ground truth, for a certain time window, namely $t \in [10, 15]$.

The different standard deviations are displayed in Table 2. For the theoretical values we have computed the mean and median, both for all signal and for $t \in [10, 15]$ for comparison with the empirical values.

We can see that the theoretical and empirical values agree quite well for the translation. The reason that the mean of the theoretical standard deviation is higher for all signal is due to the parts of the signal that are more uncertain. However, in the chosen time window the values agree well.

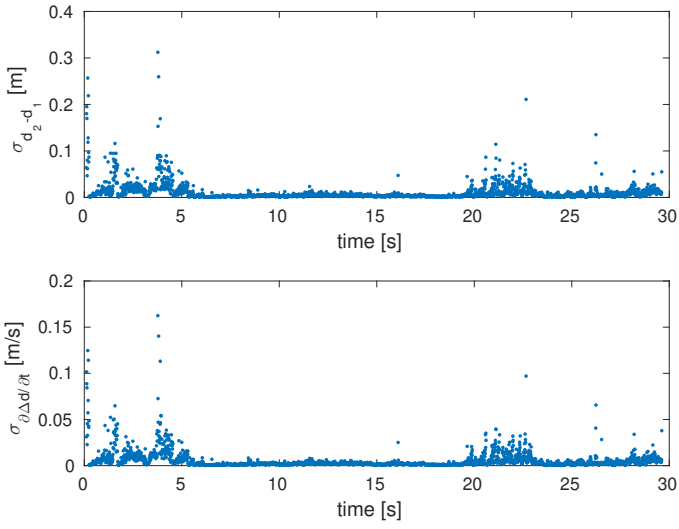


Figure 11: The standard deviation for parameters that was estimated for the real data. The upper plot shows the standard deviation of the distance difference in Figure 8 over time and the lower plot shows the standard deviation of the derivative of the distance difference in Figure 9.

For the Doppler factor the theoretical standard deviation is lower compared to the empirical estimates. This is interesting and there can be several reasons. To begin with, we made some assumptions for the received signals when we derived the equations in Section 3.1, which are probably not true for our data. E.g. in our experiments we estimated the noise in the signals as the difference between the two signals after modification. In the bottom plot of Figure 7 we see that there is still an amplitude difference between the two signals. This means that our estimated noise will not be w.s.s., as was assumed in the derivations. Furthermore, the noise will thus be overestimated. Actually, it turned out the the SNR was below 4.7.

Except from this, our method is developed to work with one signal with constant parameters and does not take into account that the patches in our real data actually constitutes one long signal. Also, we might have forgotten to take some important factor into account in out derivations for the standard deviation of the Doppler factor. It might be that the problem cannot be modeled as linear. Regardless, an interesting point for future focus is to investigate this.

4.3 Expanding the Model for Real Data

As mentioned in Section 3.2 it is in practice not much harder to estimate three model parameters. Therefore, to get a more precise solution (see Section 4.1 and the end of the

Table 2: The mean and the median for the standard deviation of the estimated distance difference (Figure 8) and the Doppler factor (Figure 9) for the two received signals.

		Translation, $d_2 - d_1$	**Doppler factor,** $\partial \Delta d / \partial t$
Theoretical, all signal	Mean of std	$1.03 \cdot 10^{-2}$	$5.25 \cdot 10^{-3}$
	Median of std	$4.71 \cdot 10^{-3}$	$2.39 \cdot 10^{-3}$
Theoretical, $t \in [10, 15]$	Mean of std	$4.58 \cdot 10^{-3}$	$2.29 \cdot 10^{-3}$
	Median of std	$4.12 \cdot 10^{-3}$	$2.08 \cdot 10^{-3}$
Empirical, $t \in [10, 15]$		$3.88 \cdot 10^{-3}$	$4.43 \cdot 10^{-1}$

previous section), we have also made experiments on the same data using (42) as model for the signals. The computations are made in the same manner as in the previous section but the error function (45) is replaced by

$$\int_t (V^{(i)}(t) - \gamma^{(i)} \bar{V}^{(i)}(\alpha^{(i)} t + \tilde{h}^{(i)} + h^{(i)}))^2 \, \mathrm{d}t, \tag{55}$$

and the optimization is performed over all three parameters, the subsample translation $h^{(i)}$, the Doppler factor $\alpha^{(i)}$ and the amplitude factor $\gamma^{(i)}$.

The results from using this model for the same signal patch as in Figure 7 can be seen in Figure 12. After moving the signals according to the estimated parameters the norm of the difference between the signals (bottom plot in the figures) has decreased with 20% when we included the amplitude factor compared to when we did not.

The plots for the translation parameter and the Doppler factor look similar to the plots in Figure 8 and 9. However, we can now make a comparison to ground truth for the amplitude factor γ as well.

The amplitude difference of the two received signals can be compared to $d_1^{(i)}$ and $d_2^{(i)}$. The amplitude estimate $\gamma^{(i)}$ is related to the quotient of the distances, $d_2^{(i)}/d_1^{(i)}$. Since the sound spreads as on the surface of a sphere, the distance quotient is proportional to the square root of the amplitude $\gamma^{(i)}$,

$$\frac{d_2^{(i)}}{d_1^{(i)}} = C \cdot \sqrt{\gamma^{(i)}}. \tag{56}$$

The unknown constant C depends on the gains of the two recording channels. For the experiment, the estimated proportionality constant was $C = 1.3$.

The distance quotient is plotted over time in Figure 13 – our estimations as dots and ground truth as a solid line. Again we see that they clearly follow the same pattern.

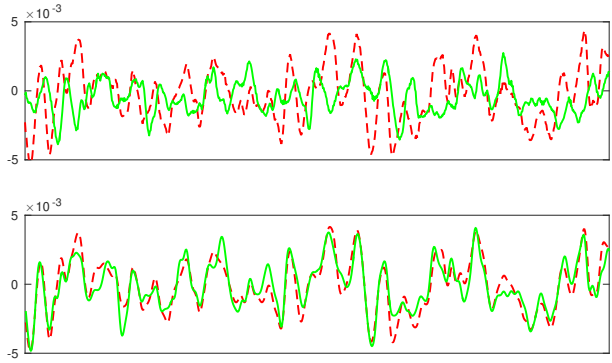


Figure 12: The plot shows the same signal patches as in Figure 7. The difference is that a larger model, namely (42), has been used here and thus an amplitude has been estimated as well. The bottom image shows the same signals after modifications using the optimal parameters.

5 Conclusions

In this paper we have studied how to estimate three parameters – time-differences, amplitude changes and minute Doppler effects – from two audio signals. The study also contains a stochastic analysis for these estimated parameters and a comparison between different signal models. The results are important both for simultaneous determination of sender and receiver positions, but also for localization, beam-forming and diarization. In the paper we have built on previous results on stochastic analysis of interpolation and smoothing in order to give explicit formulas for the covariance matrix of the estimated parameters. In the paper it is shown that the approximations that are introduced in the theory are valid as long as the smoothing is at least 0.55 sample points and as long as the signal-to-noise ratio is greater than 4.7. Furthermore, we show using experiments on both simulated and real data that these estimates provide useful information for subsequent analysis.

Acknowledgements

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

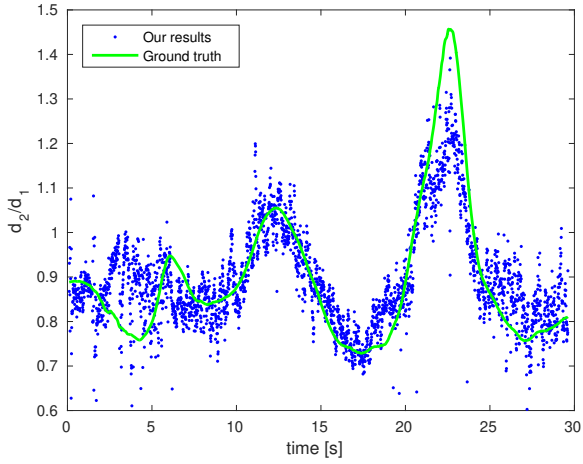


Figure 13: The distance quotient d_2/d_1 plotted over time. The solid line (—) represents the ground truth and each dot (•) is the estimation for a certain patch. While the estimations are somewhat noisy there is no doubt that the pattern is the same. The plot is similar to Figure 9 in [10], but has been generated using the updated and more independent method which is presented in this paper.

References

[1] Anguera, X., Bozonnet, S., Evans, N., Fredouille, C., Friedland, G., Vinyals, O.: Speaker diarization: A review of recent research. IEEE Transactions on Audio, Speech, and Language Processing 20(2), 356–370 (2012)

[2] Anguera, X., Wooters, C., Hernando, J.: Acoustic beamforming for speaker diarization of meetings. IEEE Transactions on Audio, Speech, and Language Processing 15(7), 2011–2022 (2007)

[3] Åström, K., Heyden, A.: Stochastic analysis of scale-space smoothing. Advances in Applied Probability 30(1) (1999)

[4] 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)

[5] Brandstein, M., Adcock, J., Silverman, H.: A closed-form location estimator for use with room environment microphone arrays. Speech and Audio Processing, IEEE Transactions on 5(1), 45 –50 (Jan 1997)

[6] Cirillo, A., Parisi, R., Uncini, A.: Sound mapping in reverberant rooms by a robust direct method. In: Acoustics, Speech and Signal Processing, IEEE International Conference on. pp. 285 –288 (April 2008)

[7] Cobos, M., Marti, A., Lopez, J.: A modified SRP-PHAT functional for robust real-time sound source localization with scalable spatial sampling. Signal Processing Letters, IEEE 18(1), 71 –74 (Jan 2011)

[8] Crocco, M., Del Bue, A., Bustreo, M., Murino, V.: A closed form solution to the microphone position self-calibration problem. In: ICASSP (March 2012)

[9] Do, H., Silverman, H., Yu, Y.: A real-time SRP-PHAT source location implementation using stochastic region contraction(SRC) on a large-aperture microphone array. In: ICASSP 2007. vol. 1, pp. 121–124 (April 2007)

[10] Flood, G., Heyden, A., Åström, K.: Estimating uncertainty in time-difference and doppler estimates. In: 7th International Conference on Pattern Recognition Applications and Methods (2018)

[11] Knapp, C., Carter, G.: The generalized correlation method for estimation of time delay. IEEE Transactions on Acoustics, Speech and Signal Processing 24(4), 320–327 (1976)

[12] Kuang, Y., Åström, K.: Stratified sensor network self-calibration from tdoa measurements. In: EUSIPCO (2013)

[13] Kuang, Y., Burgess, S., Torstensson, A., Åström, K.: A complete characterization and solution to the microphone position self-calibration problem. In: ICASSP (2013)

[14] Kuang, Y., Åström, K.: Stratified sensor network self-calibration from tdoa measurements. In: 21st European Signal Processing Conference 2013 (2013)

[15] Lindeberg, T.: Scale-space theory: A basic tool for analyzing structures at different scales. Journal of applied statistics 21(1-2), 225–270 (1994)

[16] Lindgren, G., Rootzén, H., Sandsten, M.: Stationary Stochastic Processes for Scientists and Engineers. CRC Press (2013)

[17] Petersen, K.B., Pedersen, M.S., et al.: The matrix cookbook. Technical University of Denmark 7(15), 510 (2008)

[18] Plinge, A., Jacob, F., Haeb-Umbach, R., Fink, G.A.: Acoustic microphone geometry calibration: An overview and experimental evaluation of state-of-the-art algorithms. IEEE Signal Processing Magazine 33(4), 14–29 (2016)

[19] Pollefeys, M., Nister, D.: Direct computation of sound and microphone locations from time-difference-of-arrival data. In: Proc. of ICASSP (2008)

[20] Shannon, C.E.: Communication in the presence of noise. Proceedings of the IRE 37(1), 10–21 (1949)

[21] Zhayida, S., Segerblom Rex, S., Kuang, Y., Andersson, F., Åström, K.: An Automatic System for Acoustic Microphone Geometry Calibration based on Minimal Solvers. ArXiv e-prints (Oct 2016)

[22] 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)

[23] Zhayida, S., Åström, K.: Time difference estimation with sub-sample interpolation. Journal of Signal Processing 20(6), 275–282 (2016)

Paper II

Reprinted from *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2019, K. Batstone, G. Flood, T. Beleyur, V. Larsson, H. R. Goerlitz, M. Oskarsson and K. Åström, Robust Self-Calibration of Constant Offset Time-Difference-of-Arrival, Pages 4410-4414, Copyright 2019, with permission from IEEE.

Robust Self-Calibration of Constant Offset Time-Difference-of-Arrival

Kenneth Batstone[1], Gabrielle Flood[1], Thejasvi Beleyur[3], Viktor Larsson[2], Holger R. Goerlitz[3], Magnus Oskarsson[1] and Kalle Åström[1]

[1]*Centre for Mathematical Sciences, Lund University, Lund, Sweden*
[2]*Department of Computer Science, ETH, Zurich, Switzerland*
[3]*Acoustic and Functional Ecology, Max Planck Institute for Ornithology, Seewiesen, Germany*

Abstract: In this paper we study the problem of estimating receiver and sender positions from time-difference-of-arrival measurements, assuming an unknown constant time-difference-of-arrival offset. This problem is relevant for example for repetitive sound events. In this paper it is shown that there are three minimal cases to the problem. One of these (the five receiver, five sender problem) is of particular importance. A fast solver (with run-time under 4 μs) is given. We show how this solver can be used in robust estimation algorithms, based on RANSAC, for obtaining an initial estimate followed by local optimization using a robust error norm. The system is verified on both real and synthetic data.

Keywords: time-difference-of-arrival, constant offset, RANSAC, minimal problem

1 Introduction

The problem of estimating receiver-sender node positions from measured arrival times of radio or sound signals is a key issue in different applications such as microphone array calibration, radio antenna array calibration, mapping and positioning. This field is well researched but in this paper we will focus on the anchor-free sensor network calibration both in terms of time-of-arrival measurements (TOA) and time-difference-of-arrival measurements (TDOA). For time-of-arrival the planar case of three receivers and three senders (3R/3S) was solved in [1]. For the full 3D case the over-determined problem (10R/4S) was studied in [2], where a solver for this non-minimal case was provided. There are actually

This work was partially supported by the following funding bodies: strategic research projects ELLIIT and eSSENCE, The Sten K. Johnson foundation, Swedish Foundation for Strategic Research project - Semantic Mapping and Visual Navigation for Smart Robots - (grant no. RIT15-0038), Wallenberg Artificial Intelligence, Autonomous Systems and Software Program (WASP) funded by Knut and Alice Wallenberg Foundation, Emmy Noether research (grant no. GO2029/2-1) of the Deutsche Forschungsgemeinschaft to HRG.

three minimal cases for the 3D case, namely (4R/6S), (5R/5S) and (6R/5S). A practical solver was presented in [3]. There are in general 38, 42 and 38 solutions respectively for the three different set ups. Faster solvers for these minimal cases were provided in [4].

In this paper we study the constant offset TDOA self-calibration problem. It is a problem that naturally arises e.g. when signals are emitted with a known period. As an estimation problem it lies between TOA and full TDOA. In the paper we study the minimal (5R/5S) problem and provide a fast (few μs) solver. Robust parameter estimation often use the hypothesize and test paradigm, e.g. using random sampling consensus, [5] or one of its many variants [6–8]. In these frameworks minimal solvers are important building blocks for generating model hypotheses, and we show in the paper how a minimal solver can be used for robust parameter estimation of sender positions, receiver positions and unknown offset. The system is capable of handling missing data, outliers and noise. The algorithms are tested on synthetic data as well as real data, in an office environment and in a cave. The methods are straightforward to generalize for degenerate configurations which arise if senders or receivers are restricted to a plane or to a line.

2 Time-difference-of-arrival self calibration

The problem we are considering involves m receiver positions $\mathbf{r}_i \in \mathbb{R}^3$, $i = 1, \ldots, m$, and n sender positions $\mathbf{s}_j \in \mathbb{R}^3$, $j = 1, \ldots, n$. This could for example represent the microphone positions and locations of sound emissions, respectively. Assume that the arrival time of a sound j to receiver i is t_{ij} and that the time that sound j is emitted is T_j. Multiplying the travel time $t_{ij} - T_j$ with the speed v of the signal we obtain the distance between senders and receiver,

$$v(t_{ij} - T_j) = \|\mathbf{r}_i - \mathbf{s}_j\|_2, \tag{1}$$

where $\|.\|_2$ is the l^2-norm. The speed v is throughout the paper assumed to be known and constant.

In many settings the times of emissions T_j are unknown, but regular, *e.g.*

$$T_j = k_1 j + k_0, \tag{2}$$

where the interval k_1 is known. Inserting (2) into (1) we obtain

$$v(t_{ij} - k_1 j - k_0) = \|\mathbf{r}_i - \mathbf{s}_j\|_2. \tag{3}$$

Assuming an erroneous (but regular) emission time $\tilde{T}_j = k_1 j + \tilde{k}_0$ and introducing (the measured) $z_{ij} = v(t_{ij} - \tilde{T}_j)$ and (the unknown) $o = v(k_0 - \tilde{k}_0)$ yields the following expression

$$z_{ij} = \|\mathbf{r}_i - \mathbf{s}_j\|_2 + o. \tag{4}$$

Note that this is a simplified variant of the general time-difference-of-arrival problem (see *e.g.* [9]), which allows for a different offset o for every j,

$$z_{ij} = \|\mathbf{r}_i - \mathbf{s}_j\|_2 + o_j. \tag{5}$$

Problem 1. *(Constant Offset Time-Difference-of-Arrival Self-Calibration) Given measurements* $\tilde{z}_{ij}$

$$\tilde{z}_{ij} = \|\mathbf{r}_i - \mathbf{s}_j\|_2 + o + \epsilon_{ij}, \tag{6}$$

for a subset $W \subset I$ *of all the receiver-sender index pairs* $I = \{(i,j)|i = 1,\dots m, j = 1,\dots,n\}$ *determine receiver positions* $\mathbf{r}_i$, $i = 1,\dots,m$ *and sender positions* $\mathbf{s}_j$, $j = 1,\dots,n$ *and offset* o. *Here the errors* ϵ_{ij} *are assumed to be either* **inliers**, *in which case the errors are small* ($\epsilon_{ij} \in N(0,\sigma)$) *or* **outliers**, *in which case the measurements are way off.*

Here we will use the set W_{in} for the indices (i,j) corresponding to the inlier measurements and W_{out} for the indices corresponding to the outlier set.

3 Local optimization and the low rank relaxation

If an initial estimate of the parameters $\theta_1 = \{R, S, o\}$ is given and if the set of inliers is known, then refinement of the estimate can be found by optimization methods, *e.g.* Levenberg-Marquardt (LM) [10, 11],

$$\min_{\theta_1} f(\theta_1) = \sum_{(i,j)\in W_{\text{in}}} (z_{ij} - (\|\mathbf{r}_i - \mathbf{s}_j\|_2 + o))^2. \tag{7}$$

There is an interesting relaxation to the problem, that exploits the fact that the matrix with elements $(z_{ij}-o)^2$ is rank 5, [2]. Further simplifications use the double compaction method [9]. The double compaction matrix M is defined as the matrix with elements

$$M_{ij} = (z_{ij} - o)^2 - a_i - b_j, \tag{8}$$

and it can be shown to have rank 3, i.e. $M = U^T V$, where U is of size $3 \times m$ and V is of size $3 \times n$. The relaxed problem involves a set of parameters $\theta_2 = \{U, V, b, a, o\}$. Here the constraints can be written as

$$z_{ij} = \sqrt{u_i^T v_j + a_i + b_j} + o, \tag{9}$$

where u_i denotes column i of U and v_j denotes column j of V. Refinement of parameters can be done by performing local optimization on

$$\min_{\theta_2} f(\theta_2) = \sum_{(i,j)\in W_{\text{in}}} \left(z_{ij} - (\sqrt{u_i^T v_j + a_i + b_j} + o) \right)^2. \tag{10}$$

4 Minimal problems and solvers

By counting equations and unknowns, one finds that there are three minimal problems. The first two are the symmetric case when $m = 4, n = 7$ or $m = 7, n = 4$. This case is not addressed in this paper, but we believe it to be difficult to solve. The other case is $m = n = 5$. Here, we first present a solver for the constant offset and then discuss how to solve for sender and receiver positions.

Given a 5×5 matrix, Z, with time-difference-of-arrival measurements z_{ij}, the rank 3 constraint on the double compaction matrix in (8) can be written as

$$f(o) = \det(C^T(Z - o)^{\circ 2}C) = 0, \tag{11}$$

where

$$C = \begin{pmatrix} -1 & -1 & -1 & -1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

and $^{\circ 2}$ denotes element-wise squaring (Hadamard power). Although the elements of $(Z - o)^{\circ 2}$ are of degree 2 in o, the quadratic terms cancel out after multiplication with C^T and C. Thus the elements of $C^T(Z - o)^{\circ 2}C$ are linear in o. Since the determinant is linear in each column, the determinant $f(o)$ is a polynomial of degree four in the offset o. This can be summarized as

Theorem 1. *Given time-difference-of-arrival measurements from five receivers to five senders, there are four possible offsets o, given as the roots to the fourth degree polynomial $f(o)$, counting complex roots and multiplicity of roots.*

For each solution o it is possible to generate a solution θ_2 to the relaxed problem, according to

$$b = \begin{pmatrix} (z_{11} - o)^2 & (z_{12} - o)^2 & (z_{13} - o)^2 & (z_{14} - o)^2 & (z_{15} - o)^2 \end{pmatrix},$$

$$a = \begin{pmatrix} 0 \\ (z_{21} - o)^2 - (z_{11} - o)^2 \\ (z_{31} - o)^2 - (z_{11} - o)^2 \\ (z_{41} - o)^2 - (z_{11} - o)^2 \\ (z_{51} - o)^2 - (z_{11} - o)^2 \end{pmatrix}, \tag{13}$$

$$U = \begin{pmatrix} 0 & u_2 & u_3 & u_4 & u_5 \end{pmatrix}, \tag{14}$$

Table 1: Execution times for 5×5 minimal solvers steps. Notice that the steps of calculating o and the relaxed solution is significantly faster than upgrading to the full solution

Implementation	Matlab	C++
Calculation of o	38 μs	3.7 μs
Calculation of $\theta_2 = \{U, V, a, b, o\}$	100 μs	N/A
Calculation of $\theta_1 = \{R, S, o\}$	600 ms	22 ms

$$V = \begin{pmatrix} 0 & v_2 & v_3 & v_4 & v_5 \end{pmatrix}, \tag{15}$$

where $\begin{pmatrix} u_2 & u_3 & u_4 & u_5 \end{pmatrix}^T \begin{pmatrix} v_2 & v_3 & v_4 & v_5 \end{pmatrix}$ is any rank 3 factorization of the matrix $C^T(Z - o)^{\circ 2}C$.

From a solution θ_2 to the relaxed problem it is possible to upgrade to a solution θ_1 to the original problem. This involves solving a system of polynomial equations. The procedure was first described in [3], where an algorithm for solving this was presented. Recently, a faster algorithm was presented in [4].

An efficient implementation for calculating the four solutions of the offset o given the measurements z takes 4 μs for a C++-implementation. Generating the solution θ_2 to the relaxed problem adds a few μs. However, calculating a solution θ_1 to the original problem takes another 22 ms. Thus, it is advantageous to estimate the parameters of the relaxed problem and postpone the upgrade from θ_2 to θ_1 as a final step, see Table 1.

5 Using RANSAC for five rows

We propose the use of the fast minimal solver in an hypothesize and test framework to obtain (i) a initial estimate on the offset o and (ii) an initial inlier set. The steps are described in Algorithm 1

6 Robust estimation of parameters

We use these minimal solvers with RANSAC as described in the previous section to find one or several initial estimates of the parameters θ_2 for a subset of five receivers and k senders. The solution is extended to additional rows and/or columns using robust techniques as described in [12]. During this process it is useful to keep the errors down by occasionally refining the solutions using local optimization. This has shown to reduce failures, see e.g. [13, 14]. In the proposed estimation algorithm we postpone the upgrade from θ_2 to θ_1

Algorithm 1: Offset RANSAC

1: Randomly select 5 rows and columns. Find the four solutions on o given the time-difference-of-arrival measurements.
2: For each solution o, calculate the relaxed solution $\theta_2 = \{U, V, a, b, o\}$.
3: For selected rows and for each remaining column, check for inliers according to the residuals in (10).

until we have found a good solution involving a large portion of the receiver and sender positions.

7 Experimental Validation

7.1 Minimal Solver

To test the numerical accuracy and robustness of our minimal solver we conducted an experiment using simulated data without noise. We generated a large number of instance problems (10,000) with known offsets. We then ran our solvers and compared the returned solutions with the ground truth solution. For each instance problem we recorded the distance to the closest solution. In Figure 1 the resulting histogram of the logarithm of the absolute errors are shown. As can be seen, both implementations get close to machine precision.

7.2 Experimental Setup for Real Data

We have tested our system on (i) experiments made in an office environment and (ii) experiments made at the Orlova Chuka cave, Bulgaria.

For the office experiments, 12 microphones (8x t.bone MM-1, 4x Shure SV100) were positioned around a room ($\sim 3 \times 5\ m^2$) and measured using a laser to obtain ground truth positions of the microphones with an error of $\pm 2\ mm$. The space was cleared of most the furniture to create an open space to conduct the experiment in. The sound recordings were captured using a Roland UA-1610 Sound Capture audio interface and automatically amplified. The recordings were made using the open source software Audacity 2.3.0 with a sampling frequency of 96 *kHz* on a laptop. A synthetically generated chirp was then played using a simple loudspeaker every half second for 30 *s* while moving the speaker around in the room.

For the cave experiments, 12 microphones (4x Sanken CO-100K, 8x Knowles SPU0410)

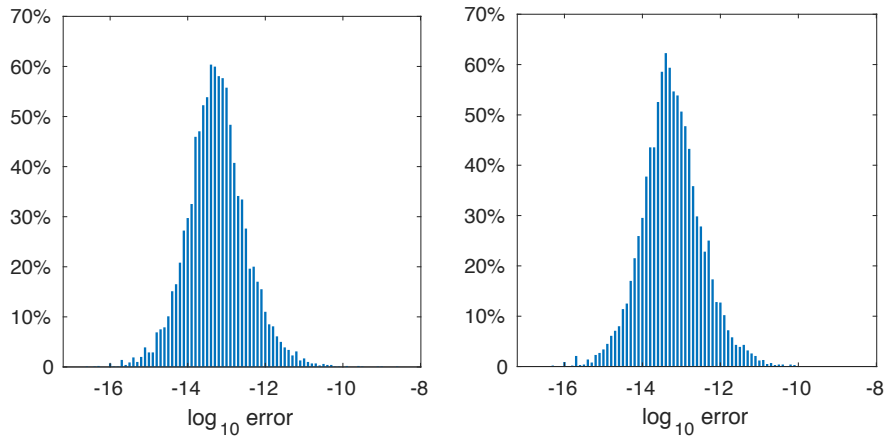


Figure 1: Left shows the histogram of the logarithm of the absolute errors, for the Matlab implementation of our minimal solver. To the right the corresponding histogram for the C++ implementation.

were positioned in a section of the cave, four microphones were placed on an inverted T array near one wall, while the other eight microphones were placed on the adjacent wall. The sound recordings were captured using pre-amplifiers (Quadmic, RME) and two synchronised Fireface 800 (RME) audio interfaces running at a sampling frequency of 192 *kHz*. Recording and playback were controlled via a custom written script based on the sound device library [15] in Python 2.7.12 [16]. Ultrasonic chirps (8 *ms*, $16 - 96$ *kHz* upward hyperbolic sweep) were played every second via one of the audio interfaces, amplified (Basetech AP-2100) and presented through a Peerless XT25SC90-04 loudspeaker. The speaker was attached to a 3-m-long pole and slowly waved in the approximately $5 \times 9 \times 3$ m^3 recording volume. Playbacks were done past 6:00 am to prevent disturbing the resident bat population.

7.3 Experimental Evaluation for Real Data

Once the office recordings were taken, an algorithm was used to find the chirps in the captured sound recordings and the algorithm then outputs the z_{ij} matrix. This can then be used in our RANSAC scheme, Algorithm 1. For this experiment we used the (5R/5S) minimal solver. A fixed number of iterations was used; 100 iterations for the initial selection of 5 receivers and senders, then the extension to more columns and rows was allowed until there was no better solution. The tolerance was set to $T = 0.01$ for the initial selection and extension of rows and column.

Once the initial values have been estimated, it underwent l^2 optimization on the inlier set. The results of the estimated microphone positions after the optimization are shown in

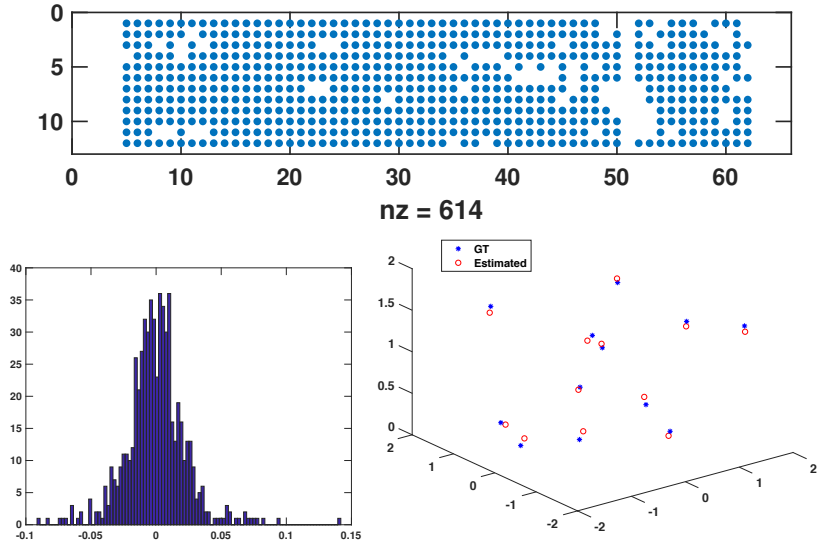


Figure 2: For the office experiment the figure shows detected inliers $\mathcal{W}_{\text{in}}$ (top), inlier residual histogram (bottom left), and estimated and ground truth microphone positions (bottom right).

Figure 2.

This produced an Euclidean distance error between each of the microphones calculated position and its ground truth position as $(0.2016, 0.0587, 0.1444, 0.1153, 0.2017, 0.1326, 0.1407, 0.1198, 0.2041, 0.2010, 0.1908, 0.2110)$ m.

For graphical purposes, a Procrustes fitting was used on the microphone positions to spread the total error over all 12 microphones. In the Procrustes fitting only rotation and translation were allowed.

For the cave experiment a similar scheme was devised and the results are shown in Figure 3.

8 Conclusions

In this paper, a novel method has been constructed to efficiently solve a TDOA problem with a constant offset. This has been verified using simulated data to test the solver and real experimental data to test our algorithms in realistic scenarios.

Looking at Figure 1 and Table 1, it can be seen that the calculation of the offsets and the calculation of the relaxed form θ_2 are very fast solvers without loss in numerical accuracy. The advantage of this is that when using a RANSAC approach, the iterations are performed quickly, giving a good initial estimate in which to optimize over, which is important in

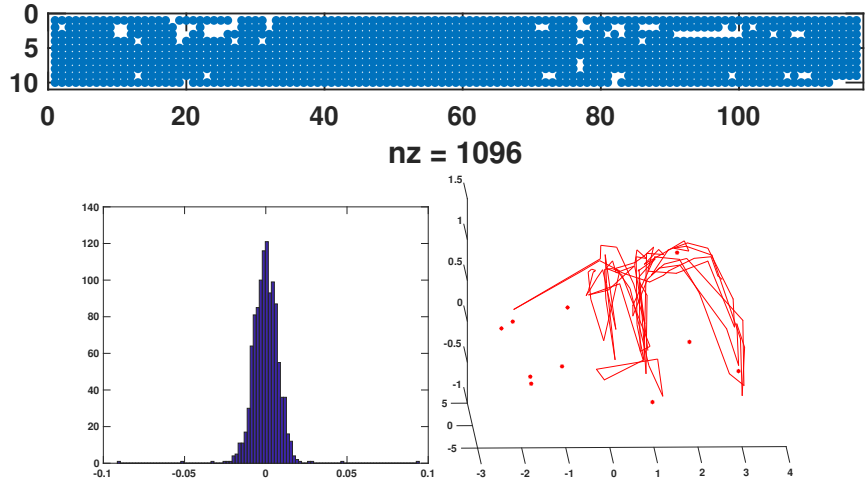


Figure 3: For the cave experiment the figure shows detected inliers $\mathcal{W}_{\text{in}}$ (top), inlier residual histogram (bottom left) and estimated microphone and sound source positions, red dots and line respectively (bottom right).

highly non-linear systems such as this.

Looking at the results from the office experiment, Figure 2, we can see that the calculated microphone positions are accurate and the residuals are small, mostly in the range $\pm 0.04\ m$. Further to this our inlier set appears to be accurate. The first and last few columns (corresponding to sound emissions) are not used in our initialisation. This is correct because the recording started before the chirps were sounded and ended after, so the chirp detection algorithm falsely determined that they were also chirps but our method decided that the data in those regions do not fit the model. A comparison of the calculated microphone positions were made to a solution from a Full TDOA system, [9], which produced similar results and very similar residuals. This provided a sanity check that the chirp detection was working correctly and that from this dataset a better solution could not be found.

For the cave experiment, similar conclusions can be made, since the residuals are very low, we can conclude that we have an accurate model. This gives a real life example of how algorithms such as the one proposed can be used.

For future work, the study of the number of inliers could be of use. At the moment our algorithm may not extend to more rows and columns if the initial solution is poor, perturbing our final solution. Perhaps a method which could adapt the initial selection in order to give a required amount of inliers could be more advantageous.

References

[1] H. Stewénius, *Gröbner Basis Methods for Minimal Problems in Computer Vision*, Ph.D. thesis, Lund University, APR 2005.

[2] M. Pollefeys and D. Nister, "Direct computation of sound and microphone locations from time-difference-of-arrival data," in *Proc. of International Conference on Acoustics, Speech and Signal Processing*, 2008.

[3] Yubin Kuang, Simon Burgess, Anna Torstensson, and Kalle Åström, "A complete characterization and solution to the microphone position self-calibration problem," in *The 38th International Conference on Acoustics, Speech, and Signal Processing*, 2013.

[4] Viktor Larsson, Kalle Åström, and Magnus Oskarsson, "Polynomial solvers for saturated ideals," in *International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[5] 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*, vol. 24, no. 6, pp. 381–95, 1981.

[6] Ondřej Chum, Jiří Matas, and Josef Kittler, "Locally optimized ransac," in *Joint Pattern Recognition Symposium*. Springer, 2003, pp. 236–243.

[7] Rahul Raguram, Ondrej Chum, Marc Pollefeys, Jiri Matas, and Jan-Michael Frahm, "Usac: a universal framework for random sample consensus.," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 2022–2038, 2013.

[8] Simon Korman and Roee Litman, "Latent ransac," in *Proc. Conf. Computer Vision and Pattern Recognition*, 2018, pp. 6693–6702.

[9] Yubin Kuang and Kalle Astrom, "Stratified sensor network self-calibration from tdoa measurements," in *Signal Processing Conference (EUSIPCO), 2013 Proceedings of the 21st European*. IEEE, 2013, pp. 1–5.

[10] Kenneth Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[11] Donald W Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[12] Kenneth Batstone, Magnus Oskarsson, and Kalle Åström, "Robust time-of-arrival self calibration with missing data and outliers," in *Signal Processing Conference (EUSIPCO), 2016 24th European*. IEEE, 2016, pp. 2370–2374.

[13] C. Engels, H. Stewenius, and D. Nister, "Bundle adjustment rules," in *PCV06*, 2006.

[14] Georg Klein and David Murray, "Parallel tracking and mapping for small ar workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

[15] M Geier, "sounddevice 0.3.5," `https://python-sounddevice.readthedocs.io/en/0.3.5/_modules/sounddevice.html`, 2015.

[16] Guido Van Rossum and Fred L Drake Jr, *Python reference manual*, Centrum voor Wiskunde en Informatica Amsterdam, 1995.

Paper III

Reprinted from *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2020, M. Larsson, G. Flood, M. Oskarsson and K. Åström, Upgrade Methods for Stratified Sensor Network Self-Calibration, Pages 4851-4855, Copyright 2020, with permission from IEEE.

Upgrade Methods for Stratified Sensor Network Self-calibration

Martin Larsson[1,2], Gabrielle Flood[1], Magnus Oskarsson[1] and Kalle Åström[1]

[1]*Centre for Mathematical Sciences, Lund University, Lund, Sweden*
[2]*Combain Mobile AB*

Abstract: Estimating receiver and sender positions is often solved using a stratified, two-tiered approach. In the first step the problem is converted to a low-rank matrix estimation problem. The second step can be seen as an affine upgrade. This affine upgrade is the focus of this paper. In the paper new efficient algorithms for solving for the upgrade parameters using minimal data are presented. It is also shown how to combine such solvers as initial estimates, either directly or after a hypothesis and test step, in optimization of likelihood. The system is verified on both real and synthetic data.

Keywords: time-of-arrival, time-difference-of-arrival, RANSAC, minimal Problems, calibration

1 Introduction

The problem of estimating receiver-sender node positions from radio or sound signals is a key issue in different applications such as microphone array calibration, radio antenna array calibration, mapping and positioning [1]. If all senders and receivers are synchronized, it is possible to obtain absolute distance measurements between senders and receivers. These measurements can be used for self-calibration and such problems (Time-of-Arrival problems, TOA) have been studied in, e.g., [2–17]. Some variants of this TOA problem are (i) TDOA - if the receivers are synchronized, whereas the senders are unsynchronized [18–20]; (ii) (COTDOA) - constant offset time-difference-of-arrival [21]; and (iii) UTDOA - if neither senders nor receivers are calibrated [22].

A popular strategy to analyze and solve these tasks is to follow a two-tiered stratified approach [18, 21, 23]. The first part of this approach is based on solving a relaxed version

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

of the problem. For this part it is possible to estimate the unknown offsets in the TDOA, COTDOA and UTDOA cases. The first part also involves estimating a low rank decomposition of a matrix derived from the measurements and offsets [24, 25]. The rank here depends on the minimum of the dimensions of the affine spans of the receivers and of the senders. Thus, for the general 3D case the rank is three, whereas the rank is two if, e.g., the receivers are coplanar and one if, e.g., the senders are colinear.

The second part of the two-tiered approach can be seen as an affine upgrade and it involves estimating a few parameters. This part is common for the cases of TOA, TDOA, COTDOA and UTDOA, since the offsets have already been estimated in the first part. Here, it is also possible to handle degenerate cases such as when the dimensions of the affine spans of the receivers and senders are different [26, 27]. The stratified approach is an efficient way of separating the full calibration problem into several well-defined sub-problems.

In this paper we study the upgrade problem, i.e., the second part of the two-tiered approach, and provide solvers for the most interesting minimal cases.[1] We also demonstrate how such solvers can be used in combination with nonlinear least squares optimization, [28, 29], to produce efficient algorithms for these upgrades. This can be used for all above mentioned stratified sensor network self-calibration problems.

2 A Stratified approach to self calibration

The problem we address involves m receiver positions $\boldsymbol{r}_i \in \mathbb{R}^3$, $i = 1, \ldots, m$, n sender positions $\boldsymbol{s}_j \in \mathbb{R}^3$, $j = 1, \ldots, n$, and possibly unknown offsets. This could for example represent the microphone positions and locations of sound emissions, respectively. The arrival time of a sound j to receiver i is denoted t_{ij} and the time that sound j is emitted is τ_j. Multiplying the travel time $t_{ij} - \tau_j$ with the speed v of the signal, we obtain the distance between senders and receivers

$$v(t_{ij} - \tau_j) = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|_2, \tag{1}$$

where $\|.\|_2$ denotes the ℓ^2-norm. The speed v is throughout the paper assumed to be known and constant. Assume that we have measurements z_{ij} of $v(t_{ij} - \tau_j)$. Then we have

$$z_{ij} = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|_2. \tag{TOA}$$

[1] The solvers were implemented in C++ and MATLAB and the code is available at `https://github.com/martinkjlarsson/upgrade-methods`.

Estimating $\boldsymbol{r}_i$ and $\boldsymbol{s}_j$ from z_{ij} is known as the node calibration problem. For the cases of COTDOA, TDOA and UTDOA the measurement equations are similar:

$$z_{ij} = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|_2 + o, \quad \text{(COTDOA)}$$
$$z_{ij} = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|_2 + o_j, \quad \text{(TDOA)}$$
$$z_{ij} = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|_2 + q_i + o_j. \quad \text{(UTDOA)}$$

The first part of the stratified approach involves estimating the offsets and will not be covered in this paper. We can thus assume that these offsets are known and subtract them from z_{ij} to get the actual distances d_{ij}. Additionally, since the distances are assumed to be positive we can without loss of generality square them and for all the cases above we get the problem

$$d_{ij}^2 = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|_2^2 = \boldsymbol{r}_i^T\boldsymbol{r}_i - 2\boldsymbol{r}_i^T\boldsymbol{s}_j + \boldsymbol{s}_j^T\boldsymbol{s}_j. \quad (2)$$

In addition to finding the offsets, the first step also includes finding a solution to the following relaxed problem

$$d_{ij}^2 = -2\boldsymbol{u}_i^T\boldsymbol{v}_j + a_j + b_i, \quad (3)$$

where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are columns of two sought matrices $\boldsymbol{U} \in \mathbb{R}^{3\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{3\times n}$, respectively, and $\boldsymbol{a}^T \in \mathbb{R}^n$ and $\boldsymbol{b} \in \mathbb{R}^m$ only depend on data. Due to noise there is typically not an exact solution to (3). Furthermore, the matrix $-2\boldsymbol{U}^T\boldsymbol{V}$ forms a low-rank approximation of the so called double compaction matrix $\boldsymbol{M}$ [18], where the rank is determined by the smallest dimension of the affine spans of the receivers and senders. For our purposes the rank will be three. The second part in the stratified approach is to take a solution to the relaxed problem and upgrade it to find the receivers and senders. This second part is the main focus of the paper.

3 Upgrade

Let $\boldsymbol{R}$ and $\boldsymbol{S}$ be the matrices whose columns are $\boldsymbol{r}_i$ and $\boldsymbol{s}_j$, respectively. The problem of upgrading a relaxed solution $(\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{a}, \boldsymbol{b})$ to a solution in $(\boldsymbol{R}, \boldsymbol{S})$ was first introduced in [11] and later improved upon in [17]. In this section we generalize the problem slightly and provide conditions which must be satisfied in the relaxed solution for the upgrade to work. We will assume that the receivers and senders are points in 3D but the scheme generalizes to any dimension. We start by introducing the matrices

$$\boldsymbol{C}_r = \boldsymbol{I} - \boldsymbol{w}_r\boldsymbol{1}^T, \quad (4)$$
$$\boldsymbol{C}_s = \boldsymbol{I} - \boldsymbol{w}_s\boldsymbol{1}^T, \quad (5)$$

where $\boldsymbol{w}_r \in \mathbb{R}^m$ and $\boldsymbol{w}_s \in \mathbb{R}^n$ are vectors such that $\boldsymbol{w}_r^T\boldsymbol{1} = \boldsymbol{w}_s^T\boldsymbol{1} = 1$. Additionally, let $\boldsymbol{r}_0 = \boldsymbol{R}\boldsymbol{w}_r$ and $\boldsymbol{s}_0 = \boldsymbol{S}\boldsymbol{w}_s$ be affine combinations of the receiver and sender positions,

respectively. The double compaction matrix, $\boldsymbol{M}$, can then be expressed in the distances, $\boldsymbol{D}$, as

$$\boldsymbol{M} = \boldsymbol{C}_r^T \boldsymbol{D}^{\circ 2} \boldsymbol{C}_s = \boldsymbol{D}^{\circ 2} - \mathbf{1}\boldsymbol{a} - \boldsymbol{b}\mathbf{1}^T + c\mathbf{1}\mathbf{1}^T \tag{6}$$

where $\boldsymbol{a} = \boldsymbol{w}_r^T \boldsymbol{D}^{\circ 2}$, $\boldsymbol{b} = \boldsymbol{D}^{\circ 2}\boldsymbol{w}_s$ and $c = \boldsymbol{w}_r^T \boldsymbol{D}^{\circ 2}\boldsymbol{w}_s$. Here $\boldsymbol{D}^{\circ 2}$ denotes the element-wise square of $\boldsymbol{D}$. By inserting (2) we get, after some simplifications,

$$\boldsymbol{M} = -2(\boldsymbol{R} - \boldsymbol{r}_0\mathbf{1}^T)^T(\boldsymbol{S} - \boldsymbol{s}_0\mathbf{1}^T), \tag{7}$$

$$a_j = \boldsymbol{w}_r^T \operatorname{diag}(\boldsymbol{R}^T\boldsymbol{R}) - 2\boldsymbol{r}_0^T\boldsymbol{s}_j + \boldsymbol{s}_j^T\boldsymbol{s}_j, \tag{8a}$$

$$b_i = \boldsymbol{r}_i^T\boldsymbol{r}_i - 2\boldsymbol{r}_i^T\boldsymbol{s}_0 + \boldsymbol{w}_s^T \operatorname{diag}(\boldsymbol{S}^T\boldsymbol{S}), \tag{8b}$$

$$c = \boldsymbol{w}_r^T \operatorname{diag}(\boldsymbol{R}^T\boldsymbol{R}) - 2\boldsymbol{r}_0^T\boldsymbol{s}_0 + \boldsymbol{w}_s^T \operatorname{diag}(\boldsymbol{S}^T\boldsymbol{S}). \tag{8c}$$

From (7) we see that, by decomposing $\boldsymbol{M} = -2\boldsymbol{U}^T\boldsymbol{V}$ using, e.g., singular value decomposition, we can find the receiver and sender positions up to some full rank transformation $\boldsymbol{L}$ and reference points $\boldsymbol{r}_0$ and $\boldsymbol{s}_0$, such that

$$\boldsymbol{R} = \boldsymbol{L}^{-T}\boldsymbol{U} + \boldsymbol{r}_0\mathbf{1}^T \quad \text{and} \quad \boldsymbol{S} = \boldsymbol{L}\boldsymbol{V} + \boldsymbol{s}_0\mathbf{1}^T. \tag{9}$$

3.1 Conditioning the relaxed problem

Due to the larger gauge freedom in the relaxed problem than in the original problem some constraints need to be added before it fits into the upgrading scheme. Firstly, we see from (9) that

$$\boldsymbol{u}_0 \triangleq \boldsymbol{U}\boldsymbol{w}_r = \boldsymbol{L}^T(\boldsymbol{R} - \boldsymbol{r}_0\mathbf{1}^T)\boldsymbol{w}_r = \mathbf{0}, \tag{10}$$

$$\boldsymbol{v}_0 \triangleq \boldsymbol{V}\boldsymbol{w}_s = \boldsymbol{L}^{-1}(\boldsymbol{S} - \boldsymbol{s}_0\mathbf{1}^T)\boldsymbol{w}_s = \mathbf{0}. \tag{11}$$

This results from our definition of $\boldsymbol{M}$ and is not true for a general $\boldsymbol{U}$ and $\boldsymbol{V}$. However, we can ensure that the conditions are met by translating $\boldsymbol{U}$ and $\boldsymbol{V}$:

$$\boldsymbol{U} \to \boldsymbol{U} - \boldsymbol{u}_0\mathbf{1}^T, \quad \boldsymbol{V} \to \boldsymbol{V} - \boldsymbol{v}_0\mathbf{1}^T \tag{12}$$

and compensating $\boldsymbol{a}$, $\boldsymbol{b}$ and c accordingly

$$\boldsymbol{a} \to \boldsymbol{a} - 2\boldsymbol{u}_0^T\boldsymbol{V}, \ \boldsymbol{b} \to \boldsymbol{b} - 2\boldsymbol{U}^T\boldsymbol{v}_0, \ c \to c - 2\boldsymbol{u}_0^T\boldsymbol{v}_0. \tag{13}$$

Secondly, from (6) we see that $c = \boldsymbol{a}\boldsymbol{w}_s = \boldsymbol{w}_r^T\boldsymbol{b}$ which can be ensured by adding appropriate constants to $\boldsymbol{a}$, $\boldsymbol{b}$ and c, making sure (3) still holds.

3.2 Solving for the upgrade parameters

We are now left with finding the unknowns $\boldsymbol{L}$, $\boldsymbol{r}_0$ and $\boldsymbol{s}_0$ using the equations in (8). Any solution to $\boldsymbol{R}$ and $\boldsymbol{S}$ is only determined up to a rigid transform. To fix the translational part of the transform we let $\boldsymbol{r}_0 = \boldsymbol{0}$. We then parameterize the remaining unknowns as $\boldsymbol{s}_0 = \boldsymbol{Lq}$ and $\boldsymbol{H} = (\boldsymbol{L}^T\boldsymbol{L})^{-1}$ where $\boldsymbol{q} \in \mathbb{R}^3$ and where $\boldsymbol{H} \in \mathbb{R}^{3\times 3}$ is a symmetric matrix.

To simplify the equations we will henceforth assume that $\boldsymbol{w}_r$ and $\boldsymbol{w}_s$ are zero vectors except for one element which is set to 1, i.e., $\boldsymbol{r}_0 = \boldsymbol{r}_i$ for some $1 \leq i \leq m$ and $\boldsymbol{s}_0 = \boldsymbol{s}_j$ for some $1 \leq j \leq n$. The equations in (8) can now be written as

$$a_j = (\boldsymbol{v}_j + \boldsymbol{q})^T \boldsymbol{H}^{-1} (\boldsymbol{v}_j + \boldsymbol{q}), \tag{14a}$$

$$b_i = \boldsymbol{u}_i^T \boldsymbol{H} \boldsymbol{u}_i - 2\boldsymbol{u}_i^T \boldsymbol{q} + \boldsymbol{q}^T \boldsymbol{H}^{-1} \boldsymbol{q}, \tag{14b}$$

$$c = \boldsymbol{q}^T \boldsymbol{H}^{-1} \boldsymbol{q}. \tag{14c}$$

$\boldsymbol{H}$ is symmetric and can together with $\boldsymbol{q}$ be parameterized in nine unknowns. However, the equations above are not independent due to the two linear constraints $c = \boldsymbol{a}\boldsymbol{w}_s = \boldsymbol{w}_r\boldsymbol{b}$. Consequently, we need $m + n + 1 \geq 11$ for the problem to be well-defined.

If we subtract c from the first two equations,

$$a_j - c = \boldsymbol{v}_j^T \boldsymbol{H}^{-1} \boldsymbol{v}_j - 2\boldsymbol{v}_j^T \boldsymbol{H}^{-1} \boldsymbol{q}, \tag{15a}$$

$$b_i - c = \boldsymbol{u}_i^T \boldsymbol{H} \boldsymbol{u}_i - 2\boldsymbol{u}_i^T \boldsymbol{q}, \tag{15b}$$

$$c = \boldsymbol{q}^T \boldsymbol{H}^{-1} \boldsymbol{q}, \tag{15c}$$

we get $m - 1$ linear constraints on the unknowns from (15b). With $m \geq 10$ the problem becomes linear and if $m < 10$, $\boldsymbol{H}$ and $\boldsymbol{q}$ can be parameterized in $10 - m$ unknowns α_k for $k = 1, \ldots, 10 - m$:

$$\boldsymbol{H} = \boldsymbol{H}_0 + \sum_{k=1}^{10-m} \alpha_k \boldsymbol{H}_k, \quad \boldsymbol{q} = \boldsymbol{q}_0 + \sum_{k=1}^{10-m} \alpha_k \boldsymbol{q}_k. \tag{16}$$

If we multiply the nonlinear equations (15a) and (15c) with $\det(\boldsymbol{H})$ they become polynomial in α_k and can be solved, e.g, using action matrix methods [9]. We might have introduced additional solutions for the case where $\det(\boldsymbol{H}) = 0$. However, these can be removed using saturation as described in [17]. Once $\boldsymbol{H}$ is solved for, $\boldsymbol{L}$ can be found using Cholesky decomposition and finally the receiver and sender positions can be attained using (9).

Table 1: The number of solutions and template sizes for the solvers, with and without saturation, together with their execution time.

	# Solutions		Template size		Exec.
Solver	no sat.	sat.	no sat.	sat.	time
900	1	-	-	-	39 s
810	3	3	-	-	130 s
801	4	4	-	-	130 s
720	9	9	12×21	12×21	170 s
711	12	12	16×28	16×28	210 s
630	21	17	88×109	112×129	600 s
621	30	26	122×152	156×182	1.2 ms
540	∞	21	-	310×331	5.3 ms
531	∞	38	-	493×531	19 ms
441	∞	42	-	817×859	72 ms

4 Minimal Solvers

In [11] two problems are considered. The first involves six receivers and four senders and results in five linear equations corresponding to (15b), three nonlinear equations corresponding to (15a) and one nonlinear equation corresponding to (15c). Because of this, we introduce a new notation and denote this problem with `531`. Similarly, the second problem in [11] involves five receivers and five senders which we would denote `441`.

Problem `531` is minimal in the sense that there are 24 distance measurements and 24 degrees of freedom in $\boldsymbol{R}$ and $\boldsymbol{S}$. However, problems `531` and `441` are both minimal in the sense that they have nine equations and nine unknowns in (15). There is indeed a total of 19 possible minimal configurations of the equations, 10 of which are listed in Table 1. The first solver, `900`, is linear and subsequent solvers get increasingly nonlinear. Note that, although we used the same scheme as in [17], our templates for setting up the action matrices for solvers `531` and `441` are slightly smaller.

5 Validation

5.1 Numerical stability of solvers

To test the numerical stability of the solvers we generated 10 random receiver and 10 random sender positions within a unit cube, $\boldsymbol{R}, \boldsymbol{S} \in [0, 1]^{3\times 10}$, from which we could calculate the distance matrix $\boldsymbol{D}$. The relaxed version of the problem ($\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{a}$, $\boldsymbol{b}$ and c) could then be found as described in Section 3. For every solver in Table 1 a minimal sample of the relaxed problem was taken and solved. From the estimated receiver and sender positions

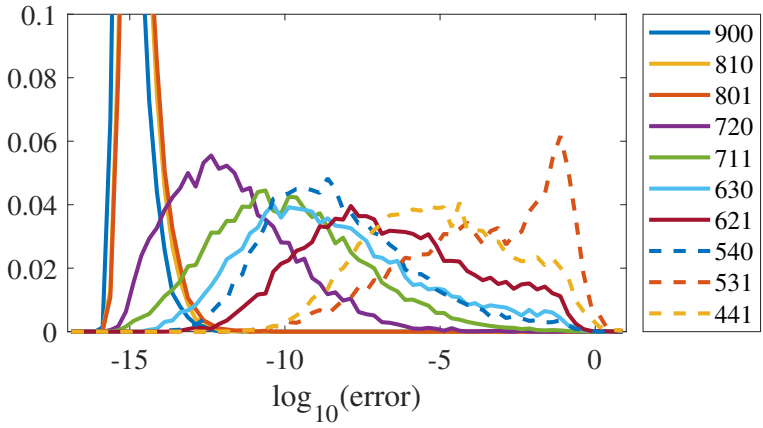


Figure 1: Error histograms for all solvers in Table 1 when provided with noise free data. Note that the graphs for 900, 810 and 801 extend beyond the plot.

the RMS error in the estimated distances was calculated. Figure 1 shows histograms of the RMS errors over multiple runs. As can be seen the 900, 810 and 801 solvers performed best and the numerical stability generally worsens as more nonlinear equations are added.

5.2 Degenerate configurations

The solvers behave differently when it comes to certain degenerate configurations of $\boldsymbol{R}$ and $\boldsymbol{S}$. For example, during testing we observed that the rank of the linear system resulting from (15b) never exceeds eight when the receivers are confined to certain two-dimensional manifolds (*e.g.* ellipsoids, paraboloids, hyperboloids). Consequently, the linear 900 solver performs poorly for such configurations. More generally, one could imagine a distance matrix $\boldsymbol{D}$ for which there are several possible embeddings of $\boldsymbol{R}$ and $\boldsymbol{S}$. If the number of embeddings exceeds the number of solutions of a solver, that solver might not find the correct embedding.

Figure 2 shows RMS distance errors for the solvers in Table 1 when the receivers and senders are located on a unit sphere. It can be seen that only solvers that include equation (15c) are stable.

5.3 Minimal solvers in a RANSAC system

In this section we show how our upgrade solvers can be used in a simple system. We assume that we have a solution to the first part in the stratified approach, i.e., we have a low rank approximation solution ($\boldsymbol{U}$, $\boldsymbol{V}$, $\boldsymbol{a}$, $\boldsymbol{b}$ and c), and want to upgrade this solution to

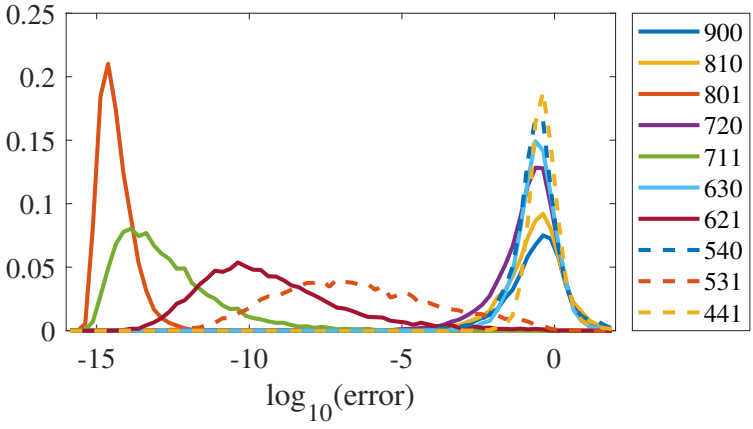


Figure 2: Error histograms for all solvers in Table 1 when provided with noise free data from points located on a sphere.

actual receiver and sender positions. From the given ($\boldsymbol{U}$, $\boldsymbol{V}$) (and a chosen minimal solver) we sample minimal configurations and solve using the chosen upgrade solver. From the solution we can estimate the distance errors. We then iterate a small number of times and choose the best solution. The results of this can be seen in Figure 3 for three example solvers. Here we have used synthetic data, and show the results for varying levels of added noise. The graph also shows the results after nonlinear optimization over the upgrade parameters $\boldsymbol{L}$ and $\boldsymbol{q}$ respectively after subsequent nonlinear optimization over the receiver and sender positions. The rationale for optimizing over $\boldsymbol{L}$ and $\boldsymbol{q}$ first, is that this is in general faster and more robust, since it only involves nine parameters. For larger problems, optimizing over the full receiver and sender positions would involve hundreds or thousands of parameters.

5.4 Real data from UWB

We evaluate the solvers on real TOA datasets gathered with an ultra-wideband (UWB) setup. Six senders were kept stationary in an area of $3 \times 3 \times 2$ meters while a receiver was moved through the setup. Ground truth positions were gathered using an optical motion capture system. The noise in the UWB measurements corresponds roughly to $\sigma = 0.26$. The RANSAC scheme discussed in the previous subsection was used to find a good initialization with three selected solvers. Table 2 shows the RMS errors in sender positions from the solver initialization, after nonlinear optimization over $\boldsymbol{L}$ and $\boldsymbol{q}$, and after nonlinear optimization over $\boldsymbol{L}$ and $\boldsymbol{q}$ followed by $\boldsymbol{R}$ and $\boldsymbol{S}$. None of the solvers performed best for all datasets and after optimization they all performed similarly.

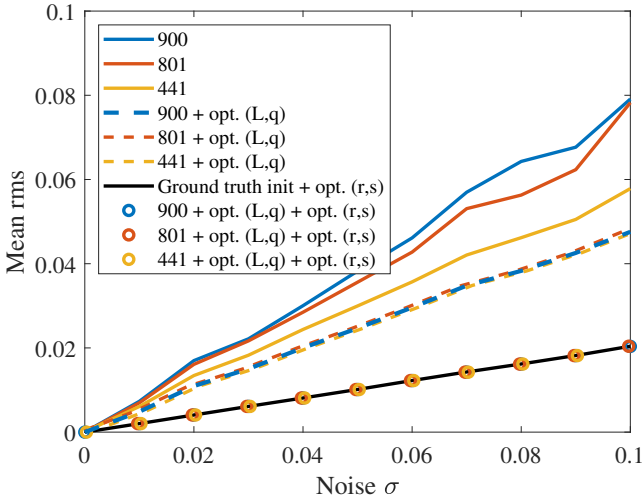


Figure 3: Results using different minimal upgrade solvers as initialization, with subsequent nonlinear optimization on synthetic data.

Table 2: RMS errors (meters) in sender positions for three real UWB datasets.

	Init			Opt. $(\boldsymbol{L}, \boldsymbol{q})$			Opt. $(\boldsymbol{L}, \boldsymbol{q})$,$(\boldsymbol{R}, \boldsymbol{S})$		
Data	900	801	441	900	801	441	900	801	441
1	0.75	1.85	1.01	0.96	0.95	0.96	0.33	0.32	0.33
2	0.62	0.73	0.48	0.38	0.38	0.38	0.28	0.28	0.28
3	0.41	0.40	0.57	0.49	0.49	0.49	0.21	0.21	0.21

6 Conclusions

In this paper, several novel solvers have been constructed to efficiently solve the upgrade step in a two-tiered stratified approach to solving TOA, TDOA and COTDOA problems. These have been verified using simulated data to test the solver and real experimental data to test our algorithms in realistic scenarios.

For future work, it would be interesting to further study how best to combine low rank estimation problems for TOA, TDOA and similar problems with affine upgrade methods. This would make it possible to produce systems that could solve a wide variety of estimation problems (TOA, TDOA, UTDOA) with a wide variety of assumptions on senders and receivers, e.g., spanning 3D or being coplanar or colinear.

References

[1] A. Plinge, F. Jacob, R. Haeb-Umbach, and G. A. Fink, "Acoustic microphone geometry calibration: An overview and experimental evaluation of state-of-the-art algorithms," *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 14–29, 2016.

[2] S. T. Birchfield and A.. Subramanya, "Microphone array position calibration by basis-point classical multidimensional scaling," *IEEE transactions on Speech and Audio Processing*, vol. 13, no. 5, 2005.

[3] D. Niculescu and B. Nath, "Ad hoc positioning system (aps)," in *Global Communications Conference (GLOBECOM)*, 2001.

[4] V. C. Raykar, I. V. Kozintsev, and R. Lienhart, "Position calibration of microphones and loudspeakers in distributed computing platforms," *IEEE transactions on Speech and Audio Processing*, vol. 13, no. 1, 2005.

[5] M. Crocco, A. Del Bue, M. Bustreo, and V. Murino, "A closed form solution to the microphone position self-calibration problem," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.

[6] J. C. Chen, R. E. Hudson, and K. Yao, "Maximum likelihood source localization and unknown sensor location estimation for wideband signals in the near-field," *IEEE transactions on Signal Processing*, vol. 50, 2002.

[7] P. Pertila, M.S. Hamalainen, and M. Mieskolainen, "Passive temporal offset estimation of multichannel recordings of an ad-hoc microphone array," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 21, no. 11, pp. 2393–2402, Nov. 2013.

[8] M. Pollefeys and D. Nister, "Direct computation of sound and microphone locations from time-difference-of-arrival data," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[9] H. Stewénius, *Gröbner Basis Methods for Minimal Problems in Computer Vision*, Ph.D. thesis, Lund University, APR 2005.

[10] Y. Kuang, E. Ask, S. Burgess, and K. Åström, "Understanding toa and tdoa network calibration using far field approximation as initial estimate," in *ICPRAM*, 2012.

[11] Y. Kuang, S. Burgess, A. Torstensson, and K. Åström, "A complete characterization and solution to the microphone position self-calibration problem," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[12] X. Li, E. Leitinger, M. Oskarsson, K. Åström, and F. Tufvesson, "Massive mimo-based localization and mapping exploiting phase information of multipath components," *IEEE Transactions on Wireless Communications*, vol. 18, no. 9, pp. 4254–4267, 2019.

[13] K. J. Batstone, M. Oskarsson, and Karl Åström, "Towards real-time time-of-arrival self-calibration using ultra-wideband anchors," in *International conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2017.

[14] S. Burgess, K. Åström, M. Högström, B. Lindquist, and R. Ljungberg, "Smartphone positioning in multi-floor environments without calibration or added infrastructure," in *International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2016.

[15] Z. Simayijiang, S. Burgess, Y. Kuang, and K. Åström, "Toa-based self-calibration of dual-microphone array," *IEEE Journal on Selected Topics in Signal Processing*, vol. 9, no. 5, pp. 791–801, 2015.

[16] Z. Simayijiang, S. Burgess, Y. Kuang, and K. Åström, "Minimal solutions for dual microphone rig self-calibration," in *European Signal Processing Conference (EURASIP)*, 2014.

[17] V. Larsson, K. Åström, and M. Oskarsson, "Polynomial solvers for saturated ideals," in *International Conference on Computer Vision (ICCV)*. IEEE, 2017.

[18] Y. Kuang and K. Astrom, "Stratified sensor network self-calibration from tdoa measurements," in *European Signal Processing Conference (EUSIPCO)*, 2013.

[19] Z. Simayijiang, S. Segerblom Rex, Y. Kuang, F. Andersson, and K. Åström, "An automatic system for acoustic microphone geometry calibration based on minimal solvers," *ARXIV*, 10 2016.

[20] L. Wang, T. Hon, J. D. Reiss, and A. Cavallaro, "Self-localization of ad-hoc arrays using time difference of arrivals," *IEEE Transactions on Signal Processing*, vol. 64, no. 4, pp. 1018–1033, Feb 2016.

[21] K. Batstone, G. Flood, T. Beleyur, V. Larsson, H. R. Goerlitz, M. Oskarsson, and K. Astrom, "Robust self-calibration of constant offset time-difference-of-arrival," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[22] S. Burgess, Y. Kuang, and K. Åström, "Node localization in unsynchronized time of arrival sensor networks," in *International Conference on Pattern Recognition (ICPR)*, 2012.

[23] K. J. Batstone, M. Oskarsson, and Karl Åström, "Robust time-of-arrival self calibration with missing data and outliers," in *European Signal Processing Conference (EUSIPCO)*, 2016.

[24] Z. Simayijiang, Fredrik Andersson, and Karl Åström, "Offset estimation for microphone localization using alternating projections," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2016.

[25] F. Jiang, Y. Kuang, and Karl Åström, "Time delay estimation for tdoa self-calibration using truncated nuclear norm regularization," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013.

[26] S. Burgess, Y. Kuang, and K. Åström, "Toa sensor network self-calibration for receiver and transmitter spaces with difference in dimension," *Signal Processing*, vol. 107, no. Online 11 June 2014, pp. 33–42, 2015.

[27] E. Ask, Y. Kuang, and K. Åström, "A unifying approach to minimal problems in collinear and planar tdoa sensor network self-calibration," in *European Signal Processing Conference (EUSIPCO)*, 2014.

[28] R. Biswas and S. Thrun, "A passive approach to sensor network localization," in *International Conference on Intelligent Robots and Systems (IROS)*, 2004.

[29] J. Wendeberg, F. Hoflinger, C. Schindelhauer, and L. Reindl, "Anchor-free tdoa self-localization," in *International conference on Indoor Positioning and Indoor Navigation (IPIN)*, 2011.

Paper IV

Reprinted from *IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP), 2021, M. Larsson, G. Flood, M. Oskarsson and K. Åström, Fast and Robust Stratified Self-Calibration Using Time-Difference-of-Arrival Measurements, Pages 4640-4644, Copyright 2021, with permission from IEEE.

Fast and Robust Stratified Self-Calibration Using Time-Difference-of-Arrival Measurements

Martin Larsson[1,2], Gabrielle Flood[1], Magnus Oskarsson[1] and Kalle Åström[1]

[1]*Centre for Mathematical Sciences, Lund University, Lund, Sweden*
[2]*Combain Mobile AB*

Abstract: In this paper we study the problem of estimating receiver and sender positions using time-difference-of-arrival measurements. For this, we use a stratified, two-tiered approach. In the first step the problem is converted to a low-rank matrix estimation problem. We present new, efficient solvers for the minimal problems of this low-rank problem. These solvers are used in a hypothesis and test manner to efficiently remove outliers and find an initial estimate which is used for the subsequent step. Once a promising solution is obtained for a sufficiently large subset of the receivers and senders, the solution can be extended to the remaining receivers and senders. These steps are then combined with robust local optimization using the initial inlier set and the initial estimate as a starting point. The proposed system is verified on both real and synthetic data.

Keywords: TDOA, self-calibration, minimal problems, RANSAC

1 Introduction

Precise localization of sender/receiver node positions using radio or sound signals is a key enabler in numerous applications such as microphone array calibration, speaker diarization, beam-forming, radio antenna array calibration, mapping and positioning [1]. There are several variants of this problem, for example (i) TOA, (ii) TDOA, (iii) COTDOA and (iv) UTDOA. The time-of-arrival (TOA) problem refers to the problem where measurements of absolute distances between senders and receivers can be obtained [2–4]. One example of this is when senders and receivers are jointly synchronized. The time-difference-of-arrival problem (TDOA) is the problem when the receivers are synchronized, whereas

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. The authors gratefully acknowledge Lund University Humanities Lab.

the senders are unsynchronized, or vice versa [5, 6]. The constant offset time-difference-of-arrival problem (COTDOA) is similar to the TDOA problem, but the unknown offset is constant [7]. Finally the UTDOA refers to the problem where neither senders nor receivers are synchronized [8].

In addition, the self-calibration problem becomes fundamentally different depending on the respective dimension of the affine hull of the senders and of the receivers. The senders and receivers can, for example, separately be confined to a line, a plane or span 3D space [9, 10].

Considering each combination of calibration type (TOA, TDOA, COTDOA, UTDOA) with each combination of sender/receiver dimensionality within a common framework is a challenge. One strategy to understand and solve the self-calibration problem is to follow a two-tiered stratified approach [6, 7]. The first part of this approach is based on solving a relaxed version of the problem where, in the case of TDOA, COTDOA and UTDOA, any offsets are solved for. The second part consists of upgrading a relaxed solution to a solution to the original problem. This was recently studied in [11].

A different approach for performing robust TDOA self-calibration was proposed in [12], where subsets of the TDOA measurements were selected to calculate candidate TOA measurements. After poor candidates were discarded the median of the remaining ones was used to perform TOA self-calibration. In [13] the redundancy of the full set of TDOA measurements was exploited using low-rank approximation to perform denoising, fill in missing data and remove outliers. However, no system for self-calibration was proposed.

In this paper we follow the stratified approach, focusing on the TDOA case in 3D. Our contribution here is twofold. First, we improve on existing minimal solvers for finding the TDOA offsets [6], making them notably faster and reducing their memory requirements. Second, utilizing these improved solvers in efficient RANSAC [14] methods we produce a system for TDOA self-calibration that is robust to noise, missing data and outliers[1]. We also verify the solvers and system using synthetic and real data.

2 Stratified Self-Calibration

The problem we address involves m receiver positions $\boldsymbol{r}_i \in \mathbb{R}^3$, $i = 1, \ldots, m$ and n sender positions $\boldsymbol{s}_j \in \mathbb{R}^3$, $j = 1, \ldots, n$. These could for example represent the microphone positions and locations of sound emissions, respectively. The arrival time of a sound j to receiver i is denoted t_{ij} and the time that sound j is emitted is denoted τ_j. Multiplying the travel time $t_{ij} - \tau_j$ with the speed v of the signal, we obtain the distance between sender and

[1]MATLAB and C++ source code can be found at `https://github.com/martinkjlarsson/tdoa-self-calibration`.

receiver

$$d_{ij} = z_{ij} - o_j = \|\boldsymbol{r}_i - \boldsymbol{s}_j\|, \tag{1}$$

where $z_{ij} = vt_{ij}$, $o_j = v\tau_j$ and $\|.\|$ denotes the ℓ^2-norm. Let $\hat{z}_{ij}$ be noisy measurements of z_{ij} that suffer from small approximately Gaussian noise, outliers with substantially larger errors and missing data. Estimating $\boldsymbol{r}_i$, $\boldsymbol{s}_j$ and o_j from $\hat{z}_{ij}$ is known as the TDOA node calibration problem.

We will use the notation $\theta_1 = \{\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{o}\}$ for the unknown parameters, where $\boldsymbol{r}_i$ and $\boldsymbol{s}_j$ are columns of $\boldsymbol{R}$ and $\boldsymbol{S}$, respectively, and $\boldsymbol{o}$ is the vector of offsets. We will also let $\hat{\boldsymbol{Z}} \in \mathbb{R}^{m\times n}$ denote the matrix with entries $\hat{z}_{ij}$ and let $\mathcal{W}_{\text{in}}$ denote the index set where $(i,j) \in \mathcal{W}_{\text{in}}$ indicates that $\hat{z}_{ij}$ is not missing and is an inlier. Given the measurements $\hat{\boldsymbol{Z}}$ and an initial solution θ_1 the refinement of the estimate can be found by local optimization methods, e.g., Levenberg-Marquardt [15, 16], by minimizing

$$f(\theta_1) = \sum_{(i,j)\in\mathcal{W}_{\text{in}}} L\left(\hat{z}_{ij} - (\|\mathbf{r}_i - \mathbf{s}_j\| + o_j)\right), \tag{2}$$

where $L(\cdot)$ is a loss function, e.g., the quadratic loss or the robust Huber loss [17].

The stratified approach is based on a relaxation of the problem, that exploits the fact that $\boldsymbol{D}^{\circ 2}$ has rank 5 [5], where $\boldsymbol{D}^{\circ 2} \in \mathbb{R}^{m\times n}$ is the matrix with entries $d_{ij}^2 = (z_{ij} - o_j)^2$. Further simplifications use the double compaction method [6]. The double compaction matrix $\boldsymbol{M} \in \mathbb{R}^{m\times n}$ is defined as the matrix with elements $M_{ij} = (z_{ij} - o_j)^2 - a_i - b_j$, where $\boldsymbol{a}$ and $\boldsymbol{b}$ are, apart from a scalar offset, affine combinations of the columns and rows of $\boldsymbol{D}^{\circ 2}$, respectively (see [11]). The matrix $\boldsymbol{M}$ can be shown to have rank 3, i.e., it can be expressed as $\boldsymbol{M} = \boldsymbol{U}^T\boldsymbol{V}$, where $\boldsymbol{U} \in \mathbb{R}^{3\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{3\times n}$. The relaxed problem thus involves a set of parameters $\theta_2 = \{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{b}, \boldsymbol{a}, \boldsymbol{o}\}$. Here the constraints can be written as $z_{ij} = \sqrt{\boldsymbol{u}_i^T\boldsymbol{v}_j + a_i + b_j} + o_j$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ denote columns i and j of $\boldsymbol{U}$ and $\boldsymbol{V}$, respectively. Refinement of the parameters can be done by local minimization of

$$f(\theta_2) = \sum_{(i,j)\in\mathcal{W}_{\text{in}}} L\left(\hat{z}_{ij} - \left(\sqrt{\boldsymbol{u}_i^T\boldsymbol{v}_j + a_i + b_j} + o_j\right)\right). \tag{3}$$

3 Minimal Solvers for the Offsets

The first step in the stratified approach is to estimate the unknown offsets $\boldsymbol{o}$. In [6] a technique for solving five minimal problems is presented (see Table 1). Throughout the paper we will use (mr/ns) to denote the problem or solver that requires m receivers and n senders. Two of the problems, (7r/4s) and (9r/5s), are linear, while the remaining three are

Table 1: Execution times and elimination template sizes for our implementation of five minimal offset solvers.

Rank	m	n	# solutions	Exec. time	Template size
2	7	4	1	37 s	14×15
2	5	6	5	810 s	37×42
3	9	5	1	36 s	29×30
3	7	6	5	1.4 ms	52×57
3	6	8	14	7.1 ms	320×334

nonlinear. In this section we propose improvements to the nonlinear ones, using automated tools from [18], making them significantly faster than in [6].

The constraint on $\boldsymbol{M}$ to be of rank 3 is equivalent to all minors of order 4 being zero. This results in a polynomial system in o_j, which can be solved using action matrix methods [19]. In the resulting solver all polynomial coefficients must be calculated from data, i.e., a large number of polynomials in z_{ij} must be evaluated. The worst solver in this regard is (6r/8s), where there are 5025 coefficients of degrees between four and eight in z_{ij}. To explicitly evaluate these one by one is time consuming and results in code that requires a lot of memory to compile if implemented in, e.g., C++.

However, using Laplace expansion the minors can be written using combinations of lower order minors and consequently, the polynomial system in $\boldsymbol{o}$ has many common subexpressions. This in turn results in common subexpressions in the coefficients which we eliminate to decrease the execution time and memory requirements of the solvers. Table 1 shows the execution time of our solvers implemented in MATLAB. These are significantly faster than the original solvers in [6], where times in the order of 500 ms was reported.

4 Minimal Solvers in RANSAC

We propose the use of the fast minimal solvers in a hypothesize and test framework to obtain (i) an initial estimate of the offsets $\boldsymbol{o}$ and (ii) an initial inlier set.

We start by randomly picking one of the minimal cases with m' receivers and n' senders. We then randomly select m' rows and n' columns of $\hat{\boldsymbol{Z}}$ containing no missing data and solve for the corresponding offsets $\boldsymbol{o}'$. For each real solution $\boldsymbol{o}'$ we can find the corresponding $\boldsymbol{a}'$, $\boldsymbol{b}'$ and double compaction matrix $\boldsymbol{M}'$. From $\boldsymbol{M}'$ we can extract $\boldsymbol{U}'$ and $\boldsymbol{V}'$ using singular value decomposition (SVD).

A partial solution can then be extended by utilizing more columns of $\hat{\boldsymbol{Z}}$. For each remaining column j we pick 5 measurements randomly from the m' selected rows and solve for $\boldsymbol{v}_j$, b_j and o_j. However, we require there to be at least 6 measurements available, so that there

are extra measurements to use for testing the extension and to classify as inliers or outliers according to the residuals in (3).

5 Systems

In this section we put the offset solvers into a robust system for solving TDOA. We use a two-tiered stratified approach as in [6] where we start by constructing a solution $\theta_2 = \{\boldsymbol{U}, \boldsymbol{V}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{o}\}$ which is later upgraded to a solution $\theta_1 = \{\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{o}\}$. Some of the components are described in detail below and the system as a whole is summarized in Algorithm 1. For comparison we also implemented a naïve system relying on random initialization, see Algorithm 2.

5.1 Initialization of θ_2

We start by initializing a solution θ_2 as described in Section 4. It is worthwhile finding a reasonable initial solution supported by many inliers, as a good initialization will speed up the remainder of the system.

5.2 Extending Solution in θ_2

The initialization will most likely yield only a partial solution in the sense that not all $\boldsymbol{u}_i$ and $\boldsymbol{v}_j$ are estimated, and that not all available measurements in $\hat{\boldsymbol{Z}}$ are used. The solution is extended with additional rows and columns using robust techniques as described in [20]. During this process it is useful to keep the errors down by occasionally refining the solution using local optimization. This has been shown to reduce failures (see e.g. [21, 22]).

5.3 Upgrade Solution in θ_2 to Solution in θ_1

A solution in θ_2 is upgraded to a solution in θ_1 using the minimal solvers presented in [11]. In RANSAC fashion a solver is randomly selected to find the upgrade parameters $\boldsymbol{L} \in \mathbb{R}^{3\times3}$ and $\boldsymbol{q} \in \mathbb{R}^3$. Receiver and sender positions are then given by the affine transformations $\boldsymbol{R} = \boldsymbol{L}^{-T}\boldsymbol{U}$ and $\boldsymbol{S} = \boldsymbol{L}(\boldsymbol{V} + \boldsymbol{q}\mathbf{1}^T)$, and inliers/outliers are classified according to the residuals in (2).

5.4 Reestimate Rows and Columns in θ_1

At this stage in the process, we can end up with receivers or senders that have not yet been estimated or that have ended up in incorrect locations, e.g., gotten stuck at a local minimum. To mitigate this, all receivers and senders are reestimated using trilateration and multilateration, respectively. If a new node position reduces the residuals in (2) it is kept, otherwise the old position is used.

Algorithm 1: Proposed system

1: Initialize solution θ_2 (Section 4).
2: Local nonlinear optimization over θ_2.
3: Extend rows and columns (Section 5.2).
4: Upgrade relaxed solution θ_2 to solution θ_1 (Section 5.3).
5: Reestimate receiver and sender positions (Section 5.4).
6: Local nonlinear optimization over θ_1 using robust norm.

Algorithm 2: Random initialization system

1: Initialize solution θ_1 randomly.
2: Local nonlinear optimization over θ_1.

6 Experimental Validation

To test the numerical accuracy and robustness of our minimal solvers we generated 10,000 synthetic problem instances with known offsets. We then ran our solvers and compared the solutions with the ground truth solution. In Figure 1 the resulting histograms of the logarithm of the errors are shown. The linear solvers, (7r/4s) and (9r/5s), performed best and overall the solvers show better numerical stability as the rank and number of solutions decrease.

Note that while the linear solvers seem to be more numerically stable and faster (see Table 1), the other solvers are still useful in scenarios where we do not have a sufficient number of receivers. One could also imagine measurements that admit multiple possible solutions to the offsets, and in those cases the linear solvers will only give one of them.

To investigate the robustness of our systems with respect to outliers and missing data we generated 15 receivers $\boldsymbol{R}$ and 100 senders $\boldsymbol{S}$ with each coordinate drawn from $\mathcal{N}(0,1)$, and corresponding offset values $\boldsymbol{o} \in \mathcal{N}(0,1)$. These were used to acquire distance measurements $\hat{\boldsymbol{Z}}$ according to (1), with additive measurement noise $\epsilon_{ij} \in \mathcal{N}(0, 0.01)$. Additionally,

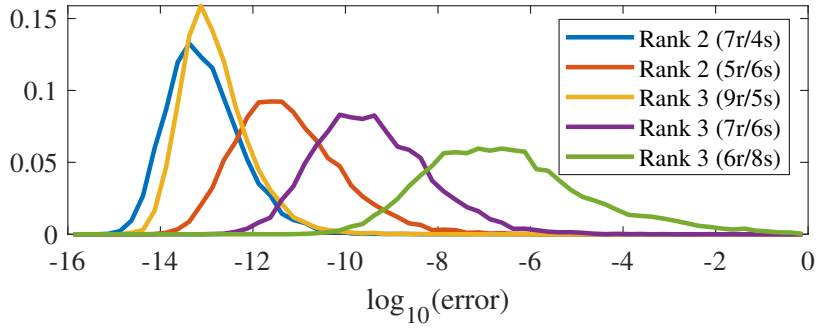


Figure 1: Histograms of the logarithm of the errors for our minimal solvers.

a number of values were deleted from $\hat{\boldsymbol{Z}}$ to simulate missing data, and some were changed to uniformly distributed values $z_{ij} \in \mathcal{U}(-2, 6)$, to simulate gross outlier measurements.

We then solved for $\theta_1 = \{\boldsymbol{R}, \boldsymbol{S}, \boldsymbol{o}\}$ using (i) Algorithm 1 and the (9r/5s) solver, (ii) Algorithm 1 and the (7r/6s) solver, (iii) Algorithm 1 and the (6r/8s) solver, and (iv) Algorithm 2. This was done 100 times to get an estimate on how often the different systems converge. A solution counted as successful if the Euclidean distance between the ground truth receiver position and the corresponding estimated receiver position was at most 0.03 for any of the receivers.

The experiment above was conducted with a fixed outlier ratio of 1 % while the ratio of missing data was varied from 0–40 %. The results from this are shown in the top plot in Figure 2. We then kept the missing data ratio fixed at 1 % while varying the ratio of outliers between 0–20 %. These results can be seen in the bottom plot of Figure 2. It is clear that our systems, from Algorithm 1, outperforms Algorithm 2. The choice of minimal solver only has a small impact on the result, but overall, the (9r/5s) solver is a better choice for the type of data synthesized here considering it is significantly faster than the other two solvers (see Table 1).

We also evaluated our system using real data. The setup consisted of 12 omni-directional microphones (the T-bone MM-1) spanning a volume of $4.0 \times 4.6 \times 1.5$ meters. A speaker was moved through the setup while emitting sound. Ground truth positions for the microphones and speaker positions where found using a Qualisys motion capture system. Seven datasets were gathered in which a chirp sound was played with regular (dataset 1-5) or irregular (dataset 6-7) intervals. The arrival times were found using cross-correlation between the recordings and the original chirp. There was no missing data. The temperature in the room was measured to be 20.1 °C which indicates a speed of sound of $v = 343\,\mathrm{m/s}$. However, we choose to disregard this estimate and consider v, and thus the scale of the solution, unknown. Because of this, the estimated microphones were registered to the ground truth

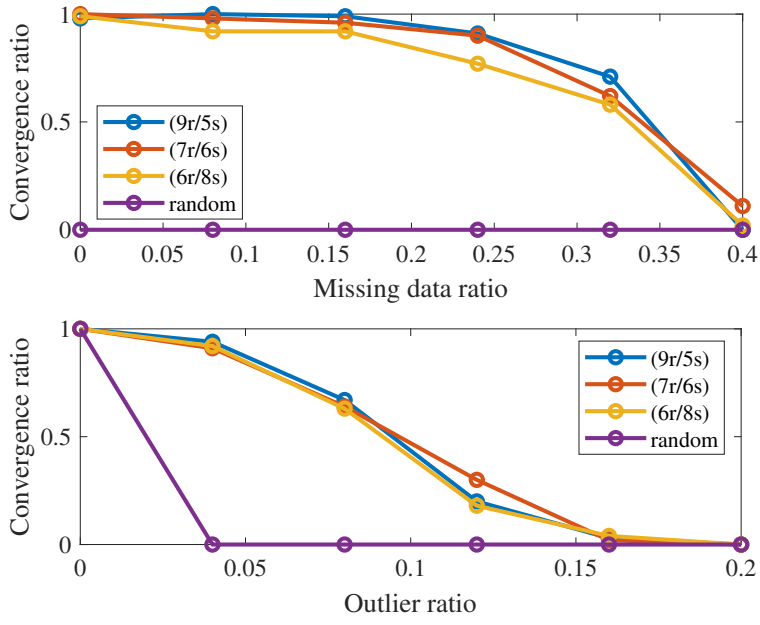


Figure 2: Convergence ratio with respect to ratio of missing data (top) and ratio of outliers (bottom) for four different systems.

using a similarity transform. The scale component of the transform was then used to estimate the speed of sound. Table 2 shows the RMS errors in the estimated microphone positions and the estimated speed of sound for each dataset. The errors are overall low with a consistent estimate of v that is close to the estimate based of the room temperature. While the true outlier rate is unknown our system classified 14–20 % of the measurements as outliers. The estimated node positions for the fourth dataset are shown in Figure 3. As can be seen, the estimated speaker positions closely follow the ground truth.

7 Conclusions

In this paper we have made several improvements to three minimal solvers for estimating offsets from time-difference-of-arrival data. The new solvers require less memory and some are two orders of magnitude faster than the state-of-the-art solvers. In the paper we also develop hypothesis and test algorithms that incorporate these new solvers and develop software systems that combine these with robust nonlinear estimation. The resulting components and systems have been tested on both synthetic and real data, where they demonstrate high quality solutions even in the presence of missing data and outliers. Consolidating further calibration types and dimensionality constraints in one coherent framework is future work.

Table 2: Results from seven real TDOA datasets including the RMSE in the estimated receiver positions.

Dataset	m	n	RMSE	Est. speed of sound
1	12	89	58 mm	347 m/s
2	12	106	50 mm	347 m/s
3	12	97	45 mm	348 m/s
4	12	105	52 mm	347 m/s
5	12	108	82 mm	348 m/s
6	12	131	64 mm	354 m/s
7	12	115	55 mm	350 m/s

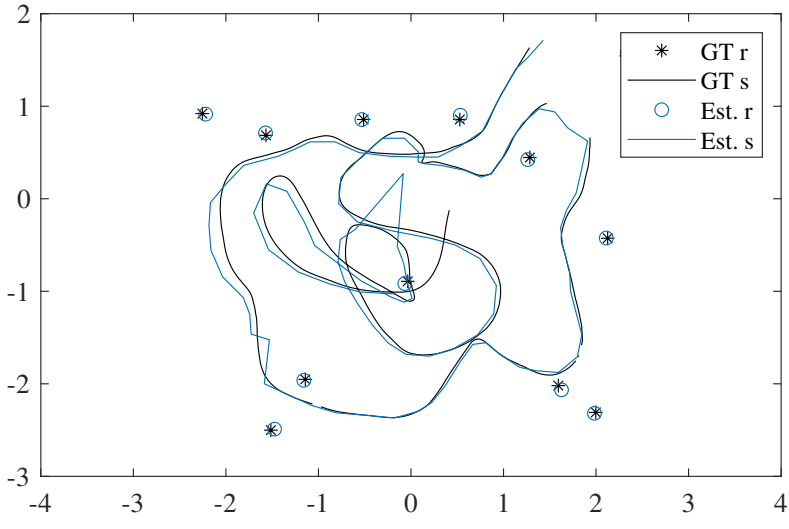


Figure 3: Top view of ground truth and estimated microphone/receiver and speaker/sender positions for the fourth dataset. The scale is in meters.

References

[1] A. Plinge, F. Jacob, R. Haeb-Umbach, and G. A. Fink, "Acoustic microphone geometry calibration: An overview and experimental evaluation of state-of-the-art algorithms," *IEEE Signal Processing Magazine*, vol. 33, no. 4, pp. 14–29, 2016.

[2] M. Crocco, A. Del Bue, M. Bustreo, and V. Murino, "A closed form solution to the microphone position self-calibration problem," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2012.

[3] H. Stewénius, *Gröbner Basis Methods for Minimal Problems in Computer Vision*, Ph.D. thesis, Lund University, 2005.

[4] Y. Kuang, S. Burgess, A. Torstensson, and K. Åström, "A complete characterization and solution to the microphone position self-calibration problem," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2013.

[5] M. Pollefeys and D. Nister, "Direct computation of sound and microphone locations from time-difference-of-arrival data," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2008.

[6] Y. Kuang and K. Åström, "Stratified sensor network self-calibration from tdoa measurements," in *European Signal Processing Conference (EUSIPCO)*, 2013.

[7] K. Batstone, G. Flood, T. Beleyur, V. Larsson, H. R. Goerlitz, M. Oskarsson, and K. Åström, "Robust self-calibration of constant offset time-difference-of-arrival," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2019.

[8] S. Burgess, Y. Kuang, and K. Åström, "Node localization in unsynchronized time of arrival sensor networks," in *International Conference on Pattern Recognition (ICPR)*, 2012.

[9] Simon Burgess, Yubin Kuang, and Kalle Åström, "Toa sensor network self-calibration for receiver and transmitter spaces with difference in dimension," *Signal Processing*, vol. 107, pp. 33–42, 2015.

[10] E. Ask, Y. Kuang, and K. Åström, "A unifying approach to minimal problems in collinear and planar tdoa sensor network self-calibration," in *European Signal Processing Conference (EUSIPCO)*, 2014.

[11] M. Larsson, G. Flood, M. Oskarsson, and K. Åström, "Robust self-calibration of constant offset time-difference-of-arrival," in *International conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020.

[12] Trung Kien Le and Nobutaka Ono, "Robust tdoa-based joint source and microphone localization in a reverberant environment using medians of acceptable recovered toas," 10 2016, Institute of Electrical and Electronics Engineers Inc.

[13] Jose Velasco, Daniel Pizarro, Javier Macias-Guarasa, and Afsaneh Asaei, "Tdoa matrices: Algebraic properties and their application to robust denoising with missing data," *IEEE Transactions on Signal Processing*, vol. 64, pp. 5242–5254, 10 2016.

[14] 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*, vol. 24, no. 6, pp. 381–95, 1981.

[15] Kenneth Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly of applied mathematics*, vol. 2, no. 2, pp. 164–168, 1944.

[16] Donald W Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the society for Industrial and Applied Mathematics*, vol. 11, no. 2, pp. 431–441, 1963.

[17] Peter J Huber, "Robust estimation of a location parameter," in *Breakthroughs in statistics*, pp. 492–518. Springer, 1992.

[18] V. Larsson, K. Åström, and M. Oskarsson, "Efficient solvers for minimal problems by syzygy-based reduction," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2383–2392.

[19] D. Cox, J. Little, and D. O'Shea, *Using Algebraic Geometry*, Springer Verlag, 1998.

[20] K. Batstone, M. Oskarsson, and K. Åström, "Robust time-of-arrival self calibration with missing data and outliers," in *2016 24th European Signal Processing Conference (EUSIPCO)*, Aug 2016, pp. 2370–2374.

[21] Chris Engels, Henrik Stewénius, and David Nistér, "Bundle adjustment rules," *Photogrammetric computer vision*, vol. 2, no. 32, 2006.

[22] Georg Klein and David Murray, "Parallel tracking and mapping for small AR workspaces," in *Mixed and Augmented Reality, 2007. ISMAR 2007. 6th IEEE and ACM International Symposium on*. IEEE, 2007, pp. 225–234.

Paper V

Paper V is not included in the electronic copy of this thesis, but can be found in the printed version.

Paper VI

Material from: G. Flood, D. Gillsjö, A. Heyden and K. Åström, Efficient Merging of Maps and Detection of Changes, *Scandinavian Conference on Image Analysis* (SCIA), 2019, published 2019, Copyright Springer Nature Switzerland AG 2019.

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, SfM

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 merging, including the problem of determining what has changed in a map. In this paper

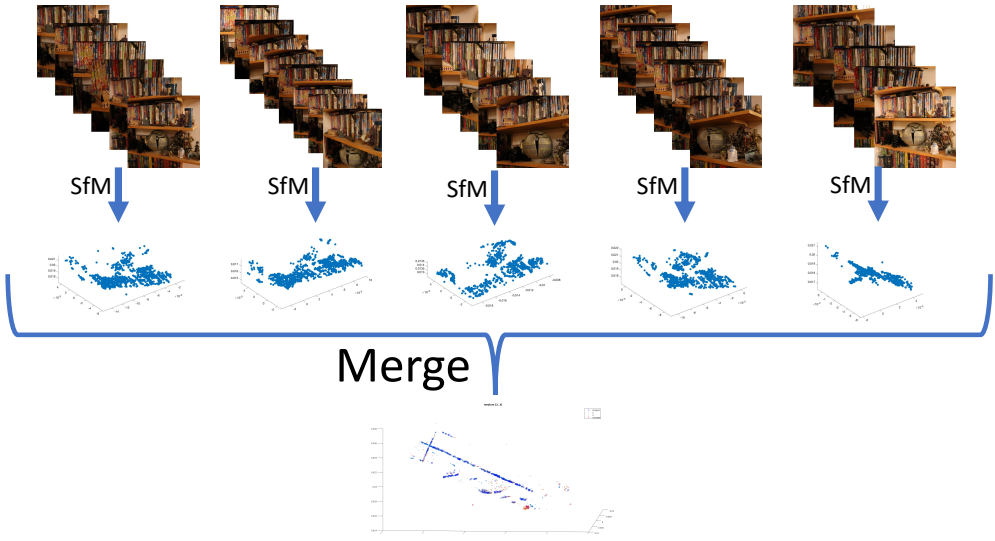


Figure 1: Structure from motion (SfM) is used to estimate a 3D map of scene features using images (or other sensors). In this paper we study the problem of detecting changes and merging maps, given multiple maps estimated by SfM from datasets collected at different occasions.

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,

$$\mathbf{r} = \begin{bmatrix} r_{11} & \dots & r_{1n} & r_{21} & \dots & r_{2n} & r_{m1} & \dots r_{mn} \end{bmatrix}^T, \qquad r_{ij} = d_{ij} - |x_i - y_j|, \tag{1}$$

and denote the parameters of interest, which are optimized, by $\mathbf{z}$. This would typically be the receiver and the sender positions,

$$\mathbf{z} = (x_1, x_2, \dots x_m, y_1, \dots y_n). \tag{2}$$

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

$$\mathbf{z} = (P_1, P_2, \dots P_m, U_1, \dots U_n). \tag{3}$$

The maximum likelihood estimate of $\mathbf{z}$ is found by minimizing the sum of the squares of the residuals, i.e.

$$\mathbf{z}^* = \text{argmin}_{\mathbf{z}} \mathbf{r}^T \mathbf{r}, \tag{4}$$

which gives the optimal parameter update

$$\Delta\mathbf{z} = -(J^T J)^{-1} J^T \mathbf{r}. \tag{5}$$

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

$$\Delta\mathbf{z} = \begin{bmatrix}\Delta\mathbf{q}\\ \Delta\mathbf{s}\end{bmatrix} = -(J^TJ)^{-1}J^T\mathbf{r} \qquad \Leftrightarrow \qquad \begin{bmatrix}U & W\\ W^T & V\end{bmatrix}\begin{bmatrix}\Delta\mathbf{q}\\ \Delta\mathbf{s}\end{bmatrix} = -J^T\mathbf{r}\,. \tag{15}$$

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

$$\mathbf{r}^T\mathbf{r} \approx (\mathbf{r}|_o + J_q|_o\Delta\mathbf{q})^T(\mathbf{r}_o + J_q|_o\Delta\mathbf{q}) = \mathbf{r}|_o^T\mathbf{r}_o + 2\mathbf{r}|_o^TJ_q|_o\Delta\mathbf{q} + \Delta\mathbf{q}^TJ_q|_o^TJ_q|_o\Delta\mathbf{q}\,. \tag{19}$$

In a minimum point $\mathbf{r}|_o^TJ_q$ is zero. Furthermore, using the QR-decomposition of the Jacobian we get

$$\Delta\mathbf{q}^TJ_q^TJ_q\Delta\mathbf{q} = \Delta\mathbf{q}^T(QR)^TQR\Delta\mathbf{q} = \Delta\mathbf{q}^TR^TQ^TQR\Delta\mathbf{q} = \Delta\mathbf{q}^TR^TR\Delta\mathbf{q}\,. \tag{20}$$

Introducing the notation $a = (\mathbf{r}|_o^T\mathbf{r}|_o)^{1/2}$, the squared residuals from (19) can be written shorter as

$$\mathbf{r}^T\mathbf{r} \approx a^2 + \Delta\mathbf{q}^TR^TR\Delta\mathbf{q}\,, \tag{21}$$

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,

$$\left(\mathbf{r}^{(i)}\right)^T\left(\mathbf{r}^{(i)}\right) = \left(a^{(i)}\right)^2 + \left(\Delta\mathbf{q}^{(i)}\right)^T\left(R^{(i)}\right)^TR^{(i)}\Delta\mathbf{q}^{(i)}\,, \qquad i = 1,2\,. \tag{22}$$

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]

$$f_{\alpha,\nu}(x) = \frac{1}{\Gamma(v)}\alpha^\nu x^{\nu-1}e^{-\alpha x}, \tag{33}$$

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 $\mathbf{q}^{(1)}, \ldots, \mathbf{q}^{(N)}$ 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.

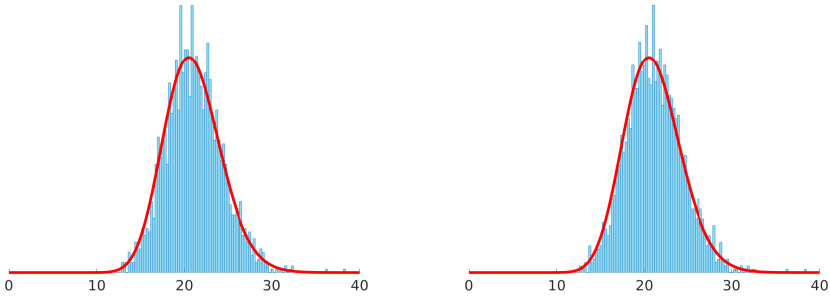


Figure 2: The plots show histograms of the residuals $\tilde{a}_{full}$ (to the left) and $\tilde{a}_{lin}$ (to the right) computed using the full bundle and the linearized method respectively. The curve (–) shows the Γ distribution which we expect $\tilde{a}$ to belong to.

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

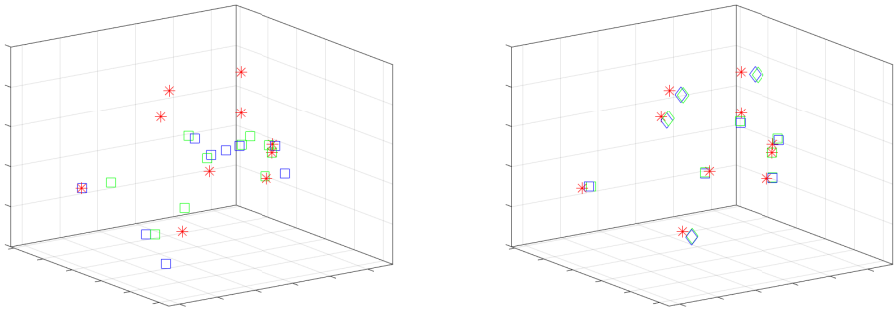


Figure 3: An unsuccessful merge of map 1,2 and 3 (left) and a successful merge of map 1 and 2 (right). The stars (∗) show the true receiver positions, the squares the results from full bundle (□) and the linearized method (□). In the right figure, the points for which a change has been detected are (correctly) 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

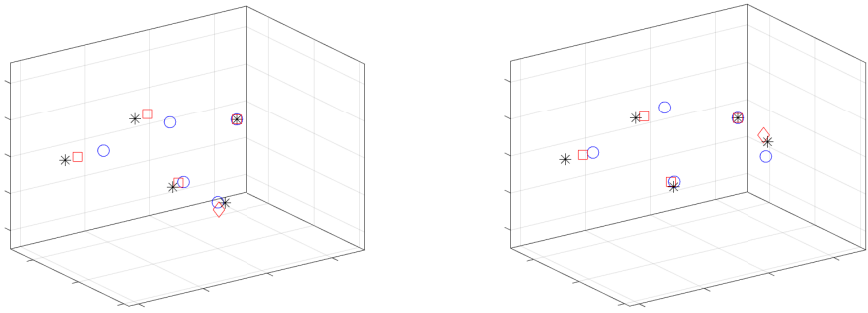


Figure 4: Results from two of the maps from the experiments with UWB data. The stars (∗) show the true receiver positions, the circles (○) the results from the Kalman filter and the squares (□) from the linearized method. The change between the maps has been correctly detected by the linearized method and changed receivers are marked with a diamond (◇).

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.

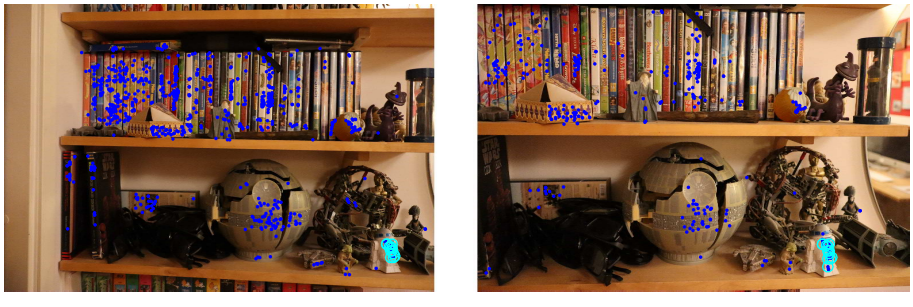


Figure 5: Changes detected in merge between dataset 2 and 3. Feature points are maked with blue dots and changed features are circled in cyan.

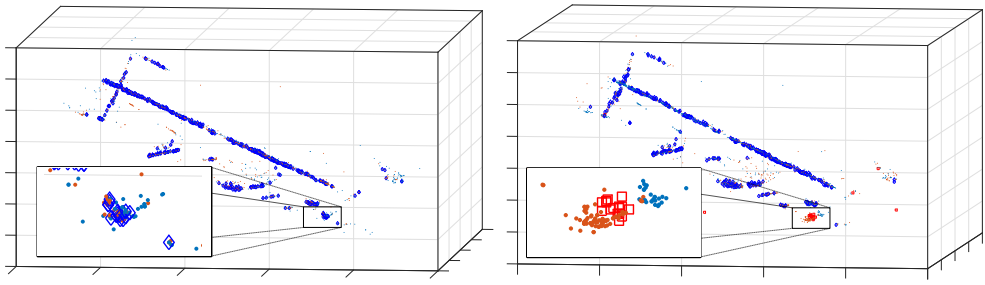


Figure 6: To the left, the merge between dataset 1 and 2 where no change was detected. The separate maps are marked with dots (●,●) and the merge by diamonds (◇). To the right, the merge between dataset 2 and 3, where a change was detected. The points for which a change was detected are marked by squares (□).

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 VII

Reprinted from *25th International Conference on Pattern Recognition* (ICPR), 2021, G. Flood, D. Gillsjö, P. Persson, A. Heyden and K. Åström, Generic Merging of Structure from Motion Maps with a Low Memory Footprint, Pages 4385-4392, Copyright 2021, with permission from IEEE.

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.

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.

Estimating map parameters and sensor motion using only sensor data is referred to as sim-

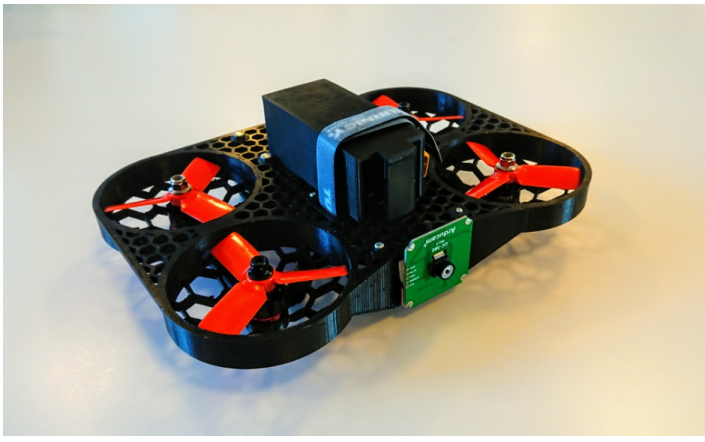


Figure 1: A drone equipped with a camera, IMU and a Raspberry Pi. The drone is used for real-time mapping and was used for one of the real data experiments in this paper.

ultaneous 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 submaps 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,

$$\mathbf{z}^* = \text{argmin}_{\mathbf{z}} \mathbf{r}^T \mathbf{r}. \tag{4}$$

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

$$\Delta \mathbf{r} = \Big(\underbrace{J_a + J_b \cdot \frac{\partial \mathbf{s}}{\partial \mathbf{q}}}_{J_q} \Big) \Delta \mathbf{q}. \tag{7}$$

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

$$\hat{\mathbf{r}} = \begin{bmatrix} a^{(1)} \\ R^{(1)}(T_1 p_1(\mathbf{q}) - \mathbf{q}^{(1)}) \\ \vdots \\ a^{(N)} \\ R^{(N)}(T_N p_N(\mathbf{q}) - \mathbf{q}^{(N)}) \end{bmatrix} \tag{11}$$

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)

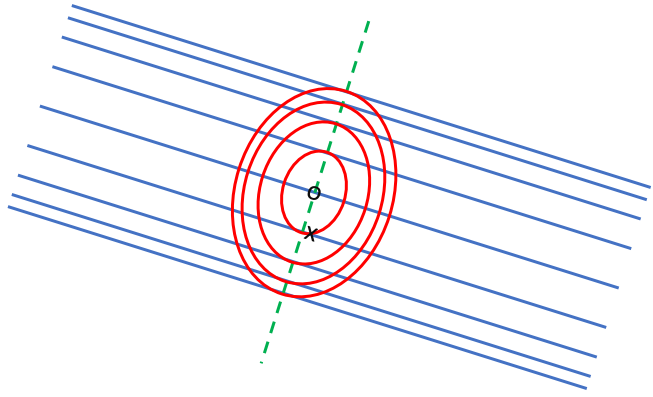


Figure 2: The blue solid lines show the level curves of the linearised error function and the o shows the point we have linearised around. By adding a penalty in the perpendicular direction, along the dashed green line, the resulting error function is the one shown as red ellipses.

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

$$\mathbf{E}[\left(a^{(k)}\right)^2] = \mathbf{E}[\left(\mathbf{r}^{(k)}\right)^T\left(\mathbf{r}^{(k)}\right)] = \sigma^2\left(\eta_{res}^{(k)} - d_{dof}^{(k)}\right). \tag{14}$$

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

$$\mathbf{E}\Big[\underbrace{\bar{a}^2 - \sum_k \left(a^{(k)}\right)^2}_{\tilde{a}}\Big] = \sigma^2\Big(\eta_{res} - d_{dof} - \sum_k \Big(\eta_{res}^{(k)} - d_{dof}^{(k)}\Big)\Big). \tag{16}$$

Letting $\tilde{a} = \bar{a}^2 - \sum_k \left(a^{(k)}\right)^2$ 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]

$$f_{\alpha,\nu}(x) = \frac{1}{\Gamma(v)}\alpha^\nu x^{\nu-1}\mathrm{e}^{-\alpha x}. \tag{18}$$

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 $\sigma = 0.05$ 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.

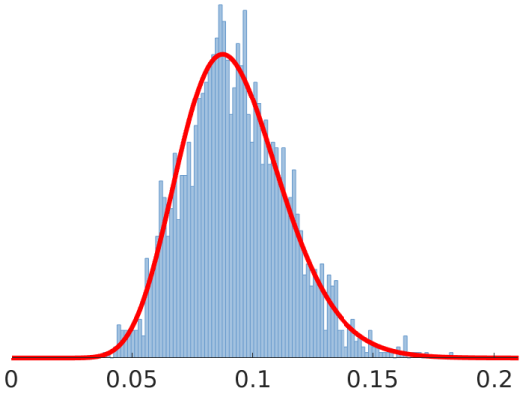


Figure 3: The image shows the histogram of $\tilde{a}$ achieved from running the same experiment several times with different noise realisations. The histogram is expected to follow the Γ distribution shown by the red curve.

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 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.

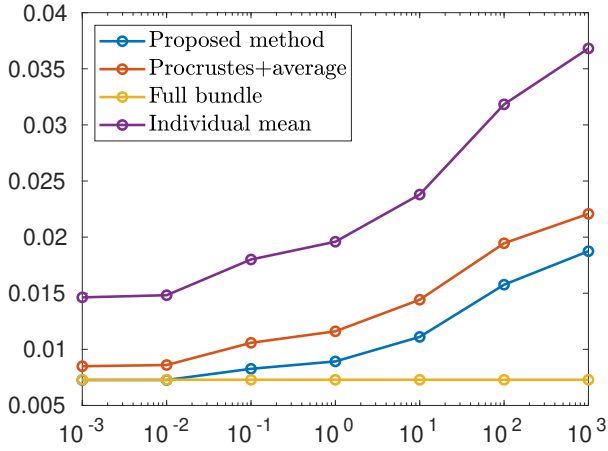


Figure 4: How the RMSE for the final map achieved using different merging methods change when the individual map bundles are terminated at different levels. The x-axis shows the Euclidean norm of the gradient and the y-axis the RMSE. The error for the individual maps is included for comparison.

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 $\sigma = 0.005$.

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

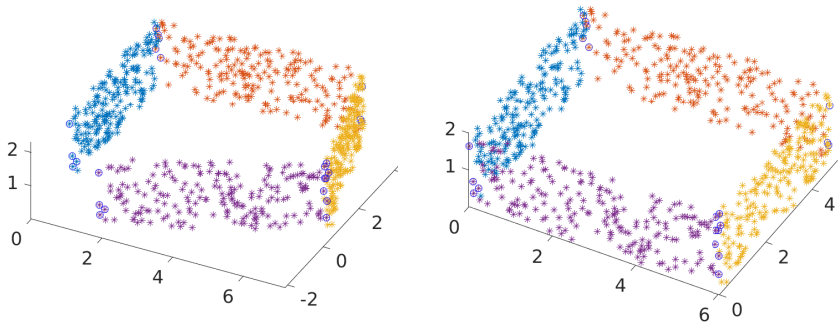


Figure 5: The left plot shows the result from merging the different sub-maps without matches in the beginning and the end of the map. To the right we have added matches between these two and the loop closure problem is solved.

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.

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

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

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 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

Figure 6: This figure shows how the R2D2 model moved between collection 2 and 3 in the bookshelf experiment.

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. 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.

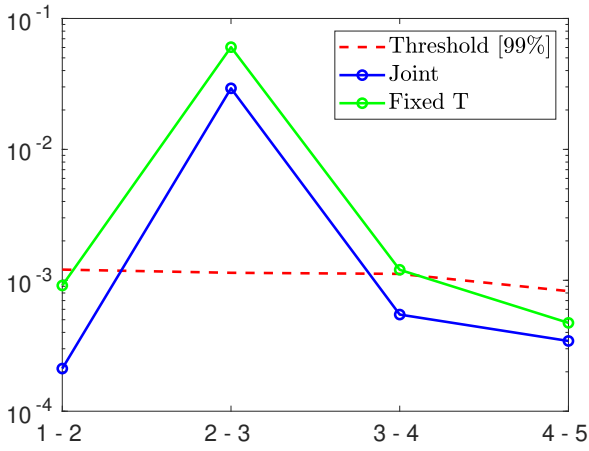


Figure 7: The sum of squared residuals for the merges of different dataset pairs. We see that jointly estimating the transform and 3D points during the merge yields smaller residuals than when estimating the transform before merging as in previous work [21]. Change between dataset 2 and 3 is correctly detected.

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

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

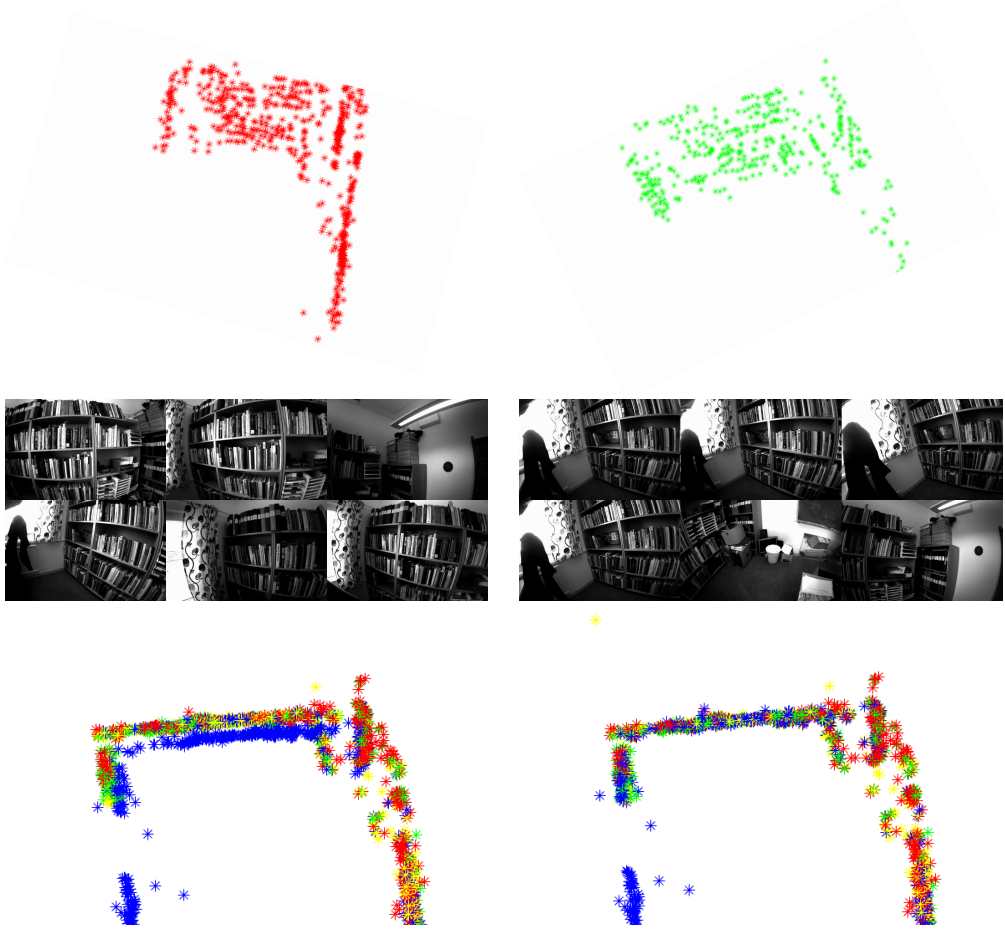


Figure 8: The two top rows shows the 3D reconstructions and a few images from two of the four drone recordings in the office experiment. The bottom row shows parts of the merged map using Procrustes to the left and our proposed method to the right. Note that the top and left walls are doubled after the Procrustes registration, while our method solves that problem.

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 VIII

Paper VIII is not included in the electronic copy of this thesis, but can be found in the printed version.

Printed by Media-Tryck, Lund 2021 NORDIC SWAN ECOLABEL 3041 0903

LUND UNIVERSITY

Doctoral Theses in Mathematical Sciences 2021:10
ISBN 978-91-8039-055-2
ISSN 1404-0034
LUTFMA-1075-2021